

# Optimization of Platform Game Levels for Player Experience

**Chris Pedersen, Julian Togelius, Georgios Yannakakis**

IT University of Copenhagen  
Rued Langgaards Vej 7, DK-2300 Copenhagen S  
Denmark  
gammabyte@gmail.com, {juto, yannakakis}@itu.dk

## Abstract

We demonstrate an approach to modelling the effects of certain parameters of platform game levels on the players' experience of the game. A version of Super Mario Bros has been adapted for generation of parameterized levels, and experiments are conducted over the web to collect data on the relationship between level design parameters and aspects of player experience. These relationships have been learned using preference learning of neural networks. The acquired models will form the basis for artificial evolution of game levels that elicit desired player emotions.

## Introduction

Numerous theories exist regarding what makes computer games fun, as well as which aspects contribute to other types of player experience (Csikszentmihalyi 1990; Koster 2005). Recently, research in player satisfaction modelling has focused on empirically measuring the effects on player experience of changing various aspects of computer games, such as NPC playing styles (Yannakakis and Hallam 2007). Such studies have been conducted using both in-game data collection, questionnaires and physiological measurements (Yannakakis and Hallam 2008a).

A parallel research direction aims to find methods for automatically generating entertaining game content. These efforts see some aspect of a game as variable, defines a fitness ("goodness") function based on a theory of player satisfaction, and uses a learning or optimization algorithm to change the variable aspect of the game so as to become more "fun" according to the definition. The few published papers on this topic deal with optimizing narrative (Nelson, Ashmore, and Mateas 2006), racing tracks (Togelius, De Nardi, and Lucas 2007), platform game levels (Compton and Mateas 2006) and game rules (Togelius and Schmidhuber 2008).

Until now, similar methodologies used for player satisfaction capture and optimization have been concentrating on the impact of NPC behavior (Yannakakis and Hallam 2007) and the adjustment of NPC internal controls for maximizing satisfaction in games (Yannakakis and Hallam 2008b). The work we describe here is concerned with the construction

---

Copyright © 2009, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.



Figure 1: Test-bed game screenshot.

of computational models of player experience derived from gameplay interaction which can be used as fitness functions for game content generation.

## Test-bed Platform game

The test-bed platform game used for our studies is a version of *Infinite Mario Bros* (see Figure 1) which is a public domain clone of Nintendo's classic platform game *Super Mario Bros*. The game is playable on the web, where Java source code is also available<sup>1</sup>. While implementing most features of the original game, its standout feature is the automatic generation of levels. Every time a new game is started, levels are randomly generated by traversing a fixed width and adding features (such as blocks, holes and enemies) according to certain heuristics. In our *Infinite Mario Bros* version most of the randomised placement of level features is fixed and deterministic since we concentrate on a few selected game level parameters that affect game experience.

## Collecting Player Data

In order to model the relation between variable features of the game and aspects of player experience, these features

---

<sup>1</sup><http://www.mojang.com/notch/mario/>

first need to be defined, and empirical data needs to be collected. We decided to focus on features which were common to most, if not all, platform games: holes and direction of movement.

### Parameterizable level generation

We modified the level generator to create levels according to four controllable parameters presented below. Three of these parameters deal with the number, width and placement of holes. The fourth parameter turns a new function, the direction switch, on or off.

- The number of holes in the level.
- The average width of holes.
- The entropy of holes with respect to which part of the level they appear in (high entropy means uniform distribution, low entropy means that they are mostly in one part of the level).
- Number of direction switches. No direction switch means that the player needs to move from left to right in order to complete the level, as in the original Super Mario Bros. If one or more direction switch is present, the level will suddenly be mirrored at random points, forcing the player to turn around and go the other way, until reaching the end of the level or the next direction switch.

Two states (low and high) for each of the four controllable parameters above are investigated. This results in  $2^4 = 16$  different variants of the game.

### Experimental methodology

We designed a game survey study to solicit pairwise emotional preferences of subjects playing different variants of the test-bed game by following the experimental protocol proposed in (Yannakakis and Hallam 2008a). Each subject plays a predefined set of four games in pairs. For each of the two pairs of games *A* and *B*, subjects report their preference for several emotional states (e.g. fun) using a 4-alternative forced choice (4-AFC) protocol.

Several statistical features are extracted from playing data which are logged during gameplay and include game completion time, time spent on various tasks (e.g. jumping, running), information on collected items (e.g. type and amount), killed enemies (e.g. type, amount, way of killing) and information on how the player died.

Data is collected over the Internet. Users are recruited via posts on blogs and mailing lists and directed to a web page containing an applet implementing the game and questionnaire<sup>2</sup>. We have so far collected enough data to have at least 2 preference instances for each pair of game variants ( $C_2^{16} = 120$  participants), but data collection is still in progress and new data will be used to improve our models.

### Modelling and optimizing player experience

We assumed there is an unknown function between individual playing characteristics (e.g. number of coins gathered),

<sup>2</sup>The game and questionnaire, which will be demonstrated at the conference, is available at [www.bluenight.dk/mario.php](http://www.bluenight.dk/mario.php)

controllable game level features (e.g. number of holes) and reported emotional preferences, and proceeded to try to predict the latter from the former.

Using neuroevolutionary preference learning of simple nonlinear perceptrons (Yannakakis and Hallam 2008a), we have been able to accurately predict certain player emotions from gameplay features. For example, whether the player is frustrated by the current game can be predicted with an accuracy of 88.66 looking at just four behavioral features. Predicting player emotions based only on controllable features is harder, but good accuracy can be achieved using nonlinear models such as multilayer perceptrons.

Work is currently in progress to use evolutionary algorithms to optimize the level design parameters (relating to holes and switches) for different objectives. We aim to be able to generate levels that tailor the playing experience according to the needs of the game design (e.g. a challenging level combined with a frustrating experience). The success of our optimization attempts will be validated with further user studies.

### Acknowledgments

The authors would like to thank Aki Järvinen and Markus Persson for insightful discussions.

### References

- Compton, K., and Mateas, M. 2006. Procedural level design for platform games. In *Proceedings of the Artificial Intelligence and Interactive Digital Entertainment International Conference (AIIDE)*.
- Csikszentmihalyi, M. 1990. *Flow: the Psychology of Optimal Experience*. Harper Collins.
- Koster, R. 2005. *A theory of fun for game design*. Paraglyph press.
- Nelson, M. J.; Ashmore, C.; and Mateas, M. 2006. Authoring an interactive narrative with declarative optimization-based drama management. In *Proceedings of the Artificial Intelligence and Interactive Digital Entertainment International Conference (AIIDE)*.
- Togelius, J., and Schmidhuber, J. 2008. An experiment in automatic game design. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games*.
- Togelius, J.; De Nardi, R.; and Lucas, S. M. 2007. Towards automatic personalised content creation in racing games. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games*.
- Yannakakis, G. N., and Hallam, J. 2007. Towards optimizing entertainment in computer games. *Applied Artificial Intelligence* 21:933–971.
- Yannakakis, G. N., and Hallam, J. 2008a. Entertainment modeling through physiology in physical play. *International Journal of Human-Computer Studies* 66:741–755.
- Yannakakis, G. N., and Hallam, J. 2008b. Real-time Adaptation of Augmented-Reality Games for Optimizing Player Satisfaction. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games*, 103–110. Perth, Australia: IEEE.

# Game and Player Feature Selection for Entertainment Capture

Georgios N. Yannakakis\* and John Hallam†

Mærsk Mc-Kinney Møller Institute

University of Southern Denmark

Campusvej 55, DK-5230, Odense

{georgios\*;john†}@mimi.sdu.dk

**Abstract**— The notion of constructing a metric of the degree to which a player enjoys a given game has been presented previously. In this paper, we attempt to construct such metric models of children's 'fun' when playing the Bug Smasher game on the Playware platform. First, a set of numerical features derived from a child's interaction with the Playware hardware is presented. Then the Sequential Forward Selection and the n-Best feature selection algorithms are employed together with a function approximator based on an artificial neural network to construct feature sets and function that model the child's notion of 'fun' for this game. Performance of the model is evaluated by the degree to which the preferences predicted by the model match those expressed by the children in a survey experiment.

The results show that an effective model can be constructed using these techniques and that the Sequential Forward Selection method performs better in this task than n-Best. The model reveals differing preferences for game parameters between children who react fast to game events and those who react slowly. The limitations and the use of the methodology as an effective adaptive mechanism to entertainment augmentation are discussed.

**Keywords:** Entertainment modeling, intelligent interactive playgrounds, neuro-evolution.

## I. INTRODUCTION

Cognitive modeling projects significant potential within digital interactive entertainment systems (such as computer games). Being able to model the level of user (gamer) engagement or satisfaction in real-time can provide insights to the appropriate AI methodology for enhancing the quality of playing experience [1] and furthermore be used to adjust digital entertainment environments according to individual user preferences.

The 'Playware' [2] intelligent interactive physical playground attempts to combine the advantages of both computer games and traditional playgrounds. On one hand, computer games keep children (among others) engaged more than other digital media because of their high degree of interactivity and the freedom for the child to develop and play a role within a fantasy world which is created during play [3]. On the other hand, traditional playgrounds offer the advantage of physical play, which furthermore improves the child's health condition, augment children's ability to engage in social and fantasy play [4], [5] and provide the freedom for children to generate their own rules for their own developed games. Experiments with children playing Playware games will be presented in this paper.

Following from the reported successful entertainment capture in physical interactive games [6], a further endeavor

on capturing player satisfaction during gameplay (i.e. entertainment modeling) through more extensive methodology and experiments with children players is presented in this paper. As in [6], this is achieved by following the theoretical principles of Malone's intrinsic qualitative factors for engaging gameplay [3], namely *challenge*, *curiosity* and *fantasy*. Quantitative measures for challenge and curiosity are used from previous studies on quantitative reported entertainment capture [6] in the Playware playground.

A mapping between the aforementioned factors (game features) and the children's notion of 'fun' or entertainment (the two terms are used interchangeably herein) is derived using a game developed on the Playware playground as a test-bed. Each player's individual characteristics (player features), such as response time and foot pressure, are recorded during each game. Feedforward artificial neural networks (ANNs) are trained using artificial evolution on this gameplay experimental data to construct a function mapping the examined game features and player features to the reported player satisfaction preferences. The n-Best (nBest) and the Sequential Forward Selection (SFS) [7] feature selection methods are used to extract the minimal subset of game and player features to be included in the ANN model.

Single feature experiments demonstrate that the average response time of children interacting with the playground is the feature that yields the best (highest-performing) mapping between game and player features and children's expressed preferences on entertainment. This result is consistent with the reported impact (i.e. significant linear correlation) of the average response time on reported entertainment in Playware games [6]. When more than one feature is examined, the SFS method generates feature subsets performing better overall than the subsets generated by the nBest method. More specifically, it finds a set of four features that yields the highest performance in matching opponent's and player's behavior to children's perceived entertainment. Player features include the player's average response time with the playground, the variance of the pressure force instances on the playground and the number of interactions with the playground. The game feature included in the most accurate model of player satisfaction obtained is the level of curiosity generated by the game opponents. Analysis of the obtained model shows that different children (classified by their response time) have different requirements on the levels of the curiosity factor for the game to be judged entertaining.

The work reported here is novel in that it isolates game and player features attributed to reported entertainment in physically demanding games and demonstrates a way of constructing a subjective model (a predictor of user preferences) of reported entertainment grounded in statistical features obtained from child-game interaction. The limitations of the proposed methodology and its extensibility to other genres of digital entertainment are discussed. Its generic use as an efficient baseline for capturing reported entertainment in physical interactive games in real-time is also outlined.

## II. ENTERTAINMENT CAPTURE

There have been several psychological studies to identify what is ‘fun’ in a game and what engages people playing computer games. Theoretical approaches include Malone’s principles of intrinsic qualitative factors for engaging game play [3], namely challenge, curiosity and fantasy as well as the well-known concepts of the theory of flow [8] incorporated in computer games as a model for evaluating player enjoyment, namely *GameFlow* [9]. A comprehensive review of the literature on qualitative approaches for modeling player enjoyment demonstrates a tendency of overlapping with Malone’s and Csikszentmihalyi’s foundational concepts. Many of these approaches are based on Lazzaro’s ‘fun’ clustering which uses four entertainment factors based on facial expressions and data obtained from game surveys on players [10]: hard fun, easy fun, altered states and socialization. Koster’s [11] theory of fun, which is primarily inspired by Lazzaro’s four factors, defines ‘fun’ as the act of mastering the game mentally. An alternative approach to fun capture is presented in [12] where fun is composed of three dimensions: durability, engagement and expectations.

Vorderer et al. [13] present a quantitative analysis of the impact of competition (i.e. challenge) on entertainment and identify challenge as the most important determinant of the enjoyment perceived by video game (*Tomb Raider*) players. They claim that a successful completion of a task generates sympathetic arousal, especially when the challenge of the task matches the player’s abilities. According to Choi et al. [14], challenge and satisfaction appear as independent processes, in contrast to the views of Malone [3] and Yannakakis et al. [6] where satisfaction derives from the appropriate level of challenge and other game components.

Iida’s work on metrics of entertainment in board games was the first attempt in the area of quantitative ‘fun’ modeling. He introduced a general metric of entertainment for variants of chess games depending on average game length and possible moves [15]. Other work in the field of quantitative entertainment capture is based on the hypothesis that the player-opponent interaction — rather than the audiovisual features, the context or the genre of the game — is the property that contributes the majority of the quality features of entertainment in a computer game [16]. Based on this fundamental assumption, a metric for measuring the real time entertainment value of predator/prey games was designed, and established as efficient and reliable by validation against human judgement [17], [18]. Further studies by Yannakakis

and Hallam [19] have shown that Artificial Neural Networks (ANN) and fuzzy neural networks can extract a better estimator of player satisfaction than a human-designed one, given appropriate estimators of the challenge and curiosity of the game and data on human players’ preferences.

A step further to entertainment capture is towards games of richer human-computer interaction and affect recognizers which are able to identify correlations between physiological signals and the human notion of entertainment. Experiments by Yannakakis et al. [20] have already shown a significant effect of children’s average heart rate on children’s reported entertainment in action games played in interactive physical playgrounds. Moreover, Rani et al. [21] propose a methodology for detecting anxiety level of the player and appropriately adjusting the level of challenge (e.g. speed) in the game of ‘Pong’. Physiological state (hear-rate, galvanic skin response) prediction models have also been proposed for potential entertainment augmentation in computer games [22]. Similar work in adjusting a game’s difficulty include endeavors through reinforcement learning [23], genetic algorithms [24], probabilistic models [25] and dynamic scripting [26]. However, the aforementioned attempts are based on the assumption that challenge is the only factor that contributes to enjoyable gaming experiences while results reported have not been cross-verified by human players.

Following the theoretical principles reported from Malone [3], Koster [11] and Yannakakis [18], this paper is primarily focused on the contributions of game opponents’ behavior to the real-time entertainment value of the game. We argue that among the three dimensions of ‘fun’ (durability, engagement, expectations) defined in [12] it is only *engagement* that is affected by the opponent since both *durability* and *expectations* are based primarily on the game design *per se*. Given a successful interactive game design that yields high expectations and durability, we only focus on the level of engagement that generates ‘fun’ (entertainment). However, instead of being based on empirical observations of children’s entertainment, the work presented here uses quantitative measures for Malone’s entertainment factors of challenge and curiosity (as introduced in [6]). On that basis, a mapping between the two aforementioned factors, children’s play recorded features and their expressed preferences is constructed using experimental data obtained from a survey experiment with children playing with Playware playground (see Section III).

## III. PLAYWARE PLAYGROUND

The Playware [2] prototype playground consists of several building blocks (i.e. tangible tiles) that allow for the game designer (e.g. the child) to develop a significant number of different games within the same platform. The overall technological concept of Playware is based on embodied AI [27] where intelligent physical identities (tiles) incorporate processing power, communication, input and output, focusing on the role of the morphology-intelligence interplay in developing game platforms. See [6], [2] for further details on Playware playground.



Fig. 1. A child playing the Bug-Smasher game.

#### A. Bug-Smasher Game

The test-bed game used for the experiments presented here is called ‘Bug-Smasher’. The game is developed on a  $6 \times 6$  square tile topology (see Fig. 1). During the game, different ‘bugs’ (colored lights) appear on the game surface and disappear sequentially after a short period of time by turning a tile’s light on and off respectively. A bug’s position is picked randomly according to the predefined level of the bugs’ spatial diversity. Spatial diversity is measured by the entropy ( $H$ ) of the bug-visited tiles.

The child’s goal is to smash as many bugs as possible by stepping on the lighted tiles. Bug-smasher has been used as a test-bed in previous work; further details can be found in [6], [20] and [28].

#### IV. EXPERIMENTAL DATA

The Bug-Smasher game has been used to acquire data of children’s judgement on entertainment. Three states (‘Low’, ‘Average’, and ‘High’) are used for each of the two entertainment factors of challenge and curiosity summing up to 9 different game states. The fantasy factor is not investigated through this survey since the focus of this paper is on the opponent (bug) contribution to entertainment. See [28] for fantasy’s positive impact on entertainment in Bug-Smasher.

We consider (as in [6]) the speed ( $S$  — in  $\text{sec}^{-1}$ ) that the bugs appear and disappear from the game and their spatial diversity ( $H$ ) on the game’s plane as appropriate measures to represent the level of challenge and the level of curiosity (unpredictability) respectively [3] during gameplay. The former provides a notion for a goal whose attainment is uncertain and the latter effectively portrays a notion of unpredictability in the subsequent events of the game — the higher the  $H$  value the higher the bug appearance unpredictability and therefore the higher the curiosity.

Seventy two normal-weighted (based on their body mass index) children whose age covered a range between 8 and 10 years participated in an experiment. By experimental design, each subject plays against two of the selected game

states in all permutations of pairs. The number of children participated in the experiment is derived from  $2 \cdot C_2^9 = 72$  being twice the required number of all combinations of 2 out of 9 game states. In this experiment, each subject plays two games ( $A$  and  $B$ ) for 90 seconds each; the two games differ in the levels of one or both entertainment factors of challenge and curiosity. Each time a pair of games is finished, the child is asked whether the first game was more ‘fun’ (see [12] for terminology used in experiments with children) than the second game i.e. whether  $A$  or  $B$  generated a more entertaining game. The 2-alternative forced choice (2-AFC) approach is used since it offers several advantages for a subjective entertainment capture: it minimizes the assumptions made about children’s notions of “fun” and allows a fair comparison between the answers of different children. Since our focus is to construct a model relating reported entertainment preferences to game and player features that generalises over the reports of different children 2-AFC is preferred to a ranking approach [29]. Note also that, children are not interviewed but are asked to fill in a questionnaire, minimizing the interviewing effects reported in [29].

The child’s answers are used to guide the training of an ANN model of reported entertainment (see Section V). In order to minimize any potential order effects we let each subject play the aforementioned games in the inverse order too. Statistical analysis of the subjects’ answers shows that no significant order effect occurs ( $r_c = -0.102$ , p-value= 0.224). The reported insignificant order effect also, in part, demonstrate that effects such as a child’s possible preference for the very first game played and the interplay between reported entertainment and familiarity with the game are statistically insignificant. The total number of game pairs played equals 144; however, data from 137 game pairs are used due to hardware (communication ports) failure during seven games.

Since, with the current implementation of the Playware platform, the only input to the system is through a Force Sensing Resistor (FSR) sensor, quantitative individual playing characteristics can only be based on three measurable features: the state (position and LEDs color) of a pressed tile, the time that a tile-press event took place and the pressure force on a pressed tile. Pressed tile events are recorded in real-time and a selection of nine personalized (individual) player features are calculated for each child. These include the number of smashed bugs over the total number of bugs appeared  $P$  (i.e. child’s score); the number of interactions with the game environment  $N_I$ ; the average and the variance of the response times ( $E\{r_t\}, \sigma^2\{r_t\}$ ); the average and the variance of the distance between the pressed tile and the bugs appearing on the game ( $E\{D_b\}, \sigma^2\{D_b\}$ ); the average and the variance of the pressure recorded from the FSR sensor  $E\{p\}, \sigma^2\{p\}$ ); and the entropy of the tiles that the child visited  $H_C$ .

#### A. Statistical Analysis

The aim of the statistical analysis presented here is to identify statistically significant correlations between children’s

notion of entertainment and any of the aforementioned individual player features and/or the quantitative entertainment factors (game features): challenge and curiosity. For this purpose the following null hypothesis is formed: The correlation between observed children judgement of entertainment and recorded player and game features, as far as the different game states are concerned, is a result of randomness. The test statistic is obtained through  $c(\vec{z}) = \sum_{i=1}^{N_s} \{z_i/N_s\}$ , where  $N_s$  is the total number of game pairs played ( $N_s = 137$ ) and  $z_i = 1$ , if the subject chooses as the more entertaining game the one with the larger value of the examined feature and  $z_i = -1$ , if the subject chooses the other game in the game pair  $i$ .

The obtained significant — significance equals 5%, high significance equals 1% in this paper — effects of the selected features on reported entertainment are:  $N_I$  ( $c(\vec{z}) = 0.1678$ , p-value = 0.0298),  $E\{r_t\}$  ( $c(\vec{z}) = -0.2262$ , p-value = 0.0050),  $\sigma^2\{r_t\}$  ( $c(\vec{z}) = -0.1532$ , p-value = 0.0435),  $E\{p\}$  ( $c(\vec{z}) = 0.1678$ , p-value = 0.0298) and  $\sigma^2\{p\}$  ( $c(\vec{z}) = 0.1970$  p-value = 0.0129). These effects appear to be commonsensical since the Bug-Smasher game belongs to the genre of action physical games where the level of engagement of the user tends to have a significant effect on the number of interactions and the reaction time of the player [30]. In Bug-Smasher, the more a child is entertained the more ( $N_I$ ) and harder ( $E\{p\}$ ) she/he tends to interact with the game platform. This behavior generates lower average response time ( $E\{r_t\}$ ) and higher average pressure on the tiles ( $E\{p\}$ ). Moreover, it appears that the variability of the aforementioned individual characteristics ( $\sigma^2\{r_t\}$ ,  $\sigma^2\{p\}$ ) does have an effect on reported entertainment too. The obtained highly significant effect of  $E\{r_t\}$  is consistent with previous experiments on the Bug-Smasher game [6].

On the other hand it appears that reported entertainment cannot be objectively modeled according to the levels of challenge ( $c(\vec{z}) = 0.0$ , p-value = 0.5382) and curiosity ( $c(\vec{z}) = 0.0909$ , p-value = 0.1448) since there exists a level of personalization which has to be included as a factor in entertainment modeling. The feature selection procedure presented in Section VI allows the designer to choose specific individual player features that can successfully map between children's behavior, game features and reported entertainment.

## V. EVOLVING ANN

The proposed approach to entertainment modeling is based on selecting a minimal subset (see Section VI) of game and player features and constructing a quantitative user model that predicts the children's reported entertainment preferences. For this purpose, a fully-connected feedforward ANN for learning the relation between the selected game and player features (ANN inputs) and the "entertainment value" (ANN output) of a game is presented. The assumption is that the entertainment value  $y$  of a given game is an unknown function of player and game features which the ANN will learn. The children's expressed preferences constrain but do not specify the values of  $y$  for individual games but we

assume that the child's expressed preferences are consistent. Since there are no prescribed target outputs for the learning problem (i.e. no differentiable output error function), ANN training algorithms such as back-propagation are inapplicable. Learning is achieved through artificial evolution [31] and is described in Section V-A.

The sigmoid function is employed at each neuron, the connection weights take values from -5 to 5 to match with input values normalized into [0, 1] before they are entered into the ANN. In an attempt to minimize the controller's size, it was determined that a single hidden-layered ANN architecture, containing 20 hidden neurons, is capable of successfully obtaining solutions of high fitness. This was determined by considering the performance of ANN architectures with up to two hidden layers containing up to 30 hidden neurons each.

### A. Genetic Algorithm

A generational genetic algorithm (GA) [32] is implemented, which uses a fitness function that measures the difference between the children's reported preferences of entertainment and the model output value  $y$ . The ANN is itself evolved. In the algorithm presented here, the ANN topology is fixed and the GA chromosome is a vector of ANN connection weights. The algorithm is described briefly in this section since it has previously presented in [6].

A population of  $N$  ( $N$  is 1000 in this paper) networks is initialized randomly. Initial real values that lie within [-5, 5] for their connection weights are picked randomly from a uniform distribution. Then, at each generation: (a) Each member (neural network) of the population is given two  $n_i$ -tuple (where  $n_i$  is the number of game or player features) values one for opponent/game  $A$  and one for opponent/game  $B$  for each pair  $j$  of games played in the survey experiment ( $N_s = 137$ ) — see [6] for further details. In each case it returns two output values, representing the level of 'fun' in each game, namely  $y_{j,A}$  and  $y_{j,B}$ . (b) Each member  $i$  of the population is evaluated via a fitness function  $f_i$  that promotes the matching between ANN outputs ( $y$ ) and children's reported answers (see [6]). A high fitness results if the ranking of  $y_{j,A}$  and  $y_{j,B}$  matches the expressed preference of the children for each game pair  $j$ . (c) A fitness-proportional selection method is used. (d) Montana and Davis [33] crossover and Gaussian mutation are applied (see [6]).

The algorithm is terminated when either a good solution is found ( $f > 0.95f_{max}$ ; where  $f_{max}$  is the maximum fitness) or a large number of generations  $g$  is completed ( $g = 10000$ ).

## VI. FEATURE SELECTION

There are two different feature selection schemes applied and compared in this paper. Given both the individual player features and the game features presented in section IV the *n Best Features Selection* (nBest) and the *Sequential Forward Selection* (SFS) methods are applied. The nBest selection method picks the  $n$  individually best features (with regards to a performance function) from the feature subset. The SFS method, by contrast, is a bottom-up search procedure where one feature is added at a time to the current feature set.

The feature to be added is selected from the subset of the remaining features so that the new feature set generates the maximum value of the performance function over all candidate features for addition [7].

The SFS method is used since it has been successfully applied in a wide variety of feature selection problems yielding high performance values with minimal feature subsets; see [34], for example, for further discussion and application to the classification problem of process identification in resistance spot welding. On the other hand, the nBest method is used for comparative purposes being the most popular technique for feature selection. Features selected by each method constitute the input vector of the evolving ANN. The feature selection procedure followed here evaluates the usability of each one of the features available and obtains the minimal feature subset that performs best in the classification between games reported as entertaining and games reported as non-entertaining (see Section V).

To evaluate the performance of each feature subset the available data is randomly divided into training and validation data sets consisting of 2/3 and 1/3 of the data respectively. The performance of an ANN model is measured through the average classification accuracy of the ANN in three independent runs using the leave-one-out cross-validation technique on the training and validation data sets. Since we are interested in the minimal feature subset that yields the highest performance we terminate the feature selection procedure (nBest or SFS) when an added feature yields equal or lower validation performance than the performance obtained without it.

#### A. Single Feature Performance

The experiment presented here tests the validation performance of single individual player and game features. Given the selected feature (ANN input), ANNs are evolved by following the approach presented in Section V-A and evaluated through the leave-one-out cross-validation method (see Section VI). The training and validation performance of each of the individual player and game features are presented in Table I where features are ranked by validation performance.

The impact of the recorded response times ( $r_t$ ) is demonstrated in Table I; both the average and the variance of these values generate the highest cross-validation performances (see also [6] for the impact of  $E\{r_t\}$  on reported entertainment in the Bug-Smasher game). Results obtained show the incapability of a single feature to successfully model reported entertainment in Bug-Smasher. Given that the best performed feature ( $E\{r_t\}$ ) yields a cross-validation performance of 62.22% it becomes apparent that more features are required to effectively model children's notion of entertainment. Moreover, it appears that results presented in Table I are consistent with the correlates of reported entertainment presented in Section IV-A. In fact, three out of four best features of Table I yield statistically significant effects on reported entertainment (see Section IV-A).

TABLE I

TRAINING AND VALIDATION PERFORMANCE OF INDIVIDUAL PLAYER AND GAME FEATURES.  $E\{r_t\}$  AND  $\sigma^2\{r_t\}$  IS THE AVERAGE AND THE VARIANCE OF THE RESPONSE TIME RESPECTIVELY;  $\sigma^2\{D_b\}$  IS THE VARIANCE OF THE DISTANCES BETWEEN THE PRESSED TILE AND THE BUGS APPEARING ON THE GAME;  $N_I$  IS THE TOTAL NUMBER OF INTERACTIONS;  $H$  IS THE QUANTITATIVE MEANS FOR THE GAME CONTROLLABLE FEATURE OF CURIOSITY;  $E\{p\}$  IS THE AVERAGE PRESSURE FORCE RECORDED FROM THE FSR SENSOR;  $H_C$  IS THE ENTROPY OF THE TILES THAT THE CHILD VISITED;  $\sigma^2\{p\}$  IS THE VARIANCE OF THE PRESSURE FORCES RECORDED FROM THE FSR SENSOR;  $E\{D_b\}$  IS THE AVERAGE DISTANCE BETWEEN THE PRESSED TILE AND THE BUGS APPEARING ON THE GAME;  $S$  IS THE QUANTITATIVE MEANS FOR THE GAME CONTROLLABLE FEATURE OF CHALLENGE AND  $P$  IS THE TOTAL NUMBER OF SMASHED BUGS.

| Feature           | Training Performance (%) | Validation Performance (%) |
|-------------------|--------------------------|----------------------------|
| $E\{r_t\}$        | 69.47                    | 62.22                      |
| $\sigma^2\{r_t\}$ | 67.91                    | 61.11                      |
| $\sigma^2\{D_b\}$ | 68.54                    | 56.67                      |
| $N_I$             | 66.36                    | 56.67                      |
| $H$               | 66.04                    | 55.56                      |
| $E\{p\}$          | 64.17                    | 53.33                      |
| $H_C$             | 59.81                    | 53.33                      |
| $\sigma^2\{p\}$   | 66.73                    | 51.11                      |
| $E\{D_b\}$        | 65.42.                   | 51.11                      |
| $S$               | 43.93                    | 46.67                      |
| $P$               | 65.42                    | 43.33                      |

#### B. More Features: Selection Method Comparison

This section presents experiments for finding the minimal feature subset that yields the highest classification performance in matching the ANNs output with children's reported answers on entertainment in unknown data (validation data set). For this purpose, the two feature selection methods described in Section VI are applied and compared. The initial subset (ANN input) for both methods includes the feature that performs best in the single feature experiment:  $E\{r_t\}$ . ANNs are evolved by following the approach presented in Section V-A. The data is partitioned in training (2/3 of total data) and validation (1/3 of total data) portions and the leave-one-out cross-validation technique is used to obtain the classification performance of the ANNs.

Table II presents the above-mentioned comparative study between nBest and SFS. SFS appears to generate feature subsets that yield higher validation performance than feature subsets generated by nBest. The best cross-validation performance (77.77%; average of 70%, 73.33% and 90%) is achieved when the ANN input contains  $E\{r_t\}$ ,  $\sigma^2\{p\}$ ,  $H$  and  $N_I$ . The binomial-distributed probability of this performance to occur at random is 0.0019 demonstrating statistical significance and providing evidence for this solution's robustness. Note that, challenge is absent from the obtained feature subset indicating that the spatial diversity of the bugs (curiosity) has a higher impact on children's reported

entertainment than the speed of the game (challenge).

Difficulties in obtaining higher classification accuracy are found in experimental noise in both the recorded features and the children's answers on self reports. Even though comparative fun analysis is a reliable and established method for capturing reported entertainment in computer [18] and mixed-reality [6] games, it generates a significant amount of uncertainty in subjects' reported answers. Uncertainty appears when the two games played are not significantly different with regards to the entertainment value they generate for the player and therefore cannot be distinguished.

TABLE II

VALIDATION PERFORMANCE OF INDIVIDUAL PLAYER AND GAME FEATURES AND THEIR RESPECTIVE BINOMIAL DISTRIBUTED P-VALUES.

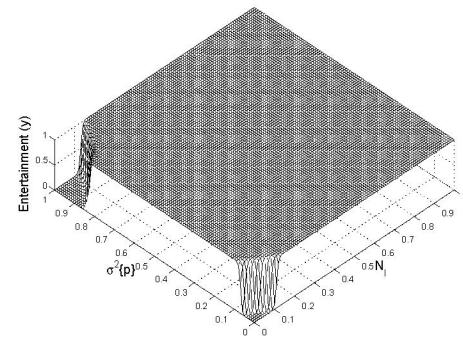
$E\{r_t\}$  AND  $\sigma^2\{r_t\}$  IS THE AVERAGE AND THE VARIANCE OF THE RESPONSE TIME RESPECTIVELY;  $\sigma^2\{p\}$  IS THE VARIANCE OF THE PRESSURE FORCES RECORDED FROM THE FSR SENSOR;  $\sigma^2\{D_b\}$  IS THE VARIANCE OF THE DISTANCES BETWEEN THE PRESSED TILE AND THE BUGS APPEARING ON THE GAME;  $H$  IS THE QUANTITATIVE METRIC OF CURIOSITY AND  $N_I$  IS THE TOTAL NUMBER OF INTERACTIONS.

| Features          | nBest | $p_{nBest}$ | Features          | SFS          | $p_{SF}$      |
|-------------------|-------|-------------|-------------------|--------------|---------------|
| $E\{r_t\}$        | 62.22 | 0.1270      | $E\{r_t\}$        | 62.22        | 0.1270        |
| $\sigma^2\{r_t\}$ | 58.88 | 0.2179      | $\sigma^2\{p\}$   | 67.77        | 0.0400        |
| $\sigma^2\{D_b\}$ | 44.44 | 0.2551      | $H$               | 68.88        | 0.0307        |
| $N_I$             | 46.67 | 0.4277      | $N_I$             | <b>77.77</b> | <b>0.0019</b> |
| $H$               | 52.22 | 0.4759      | $\sigma^2\{r_t\}$ | 63.33        | 0.1002        |

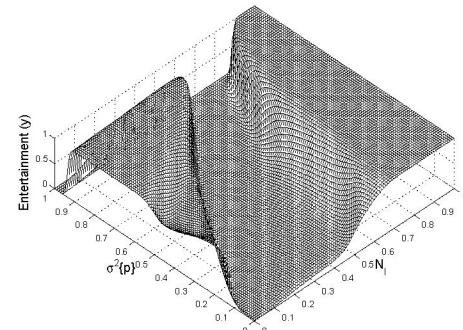
For reasons of space, only the feature subset  $\{E\{r_t\}, \sigma^2\{p\}, N_I, H\}$  with the highest validation performance (90.00%, in one of the three learning attempts) is presented in this paper. Note that, the qualitative features of the surfaces plotted in Fig. 2 appeared in all three different learning attempts of the cross-validation procedure for this feature subset.

Fig. 2 illustrates the trained ANN output with regards to  $\sigma^2\{p\}$  and  $N_I$  for six points in the  $(E\{r_t\}, H)$  search space. These values constitute the combinations of two  $E\{r_t\}$  states (0 and 1 named *Fast* and *Slow* respectively), and the three states used for  $H$  (0.33, 0.66 and 1 named *Low*, *Average* and *High* respectively). The above presentation helps towards interpreting the mapping between  $\sigma^2\{p\}$ ,  $N_I$  and reported entertainment according to how fast children react with the playground and the level of curiosity.

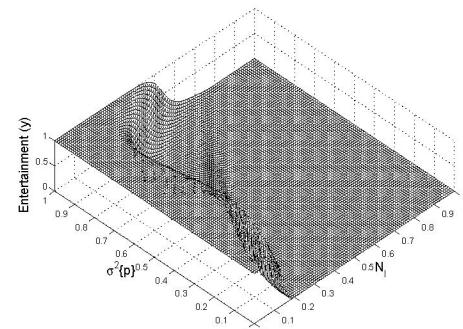
As seen from Fig. 2, fast children ( $E\{r_t\} = 0$ ) appear to enjoy average and high curiosity values except when high  $N_I$  values are combined with low values of  $\sigma^2\{p\}$  (see Fig. 2(e) and Fig. 2(f)). Fast children's preference for low levels of curiosity is met only when their behavior combines low values of  $N_I$  and high values of  $\sigma^2\{p\}$  (see Fig. 2(d)). On the other hand, slow children appear to prefer low curiosity levels except when the  $N_I$  value they generate is low and combined with either very high or very low  $\sigma^2\{p\}$  values (see Fig. 2(a)). Average curiosity levels are preferred by slow children in many fewer cases; that is when their  $N_I$  value



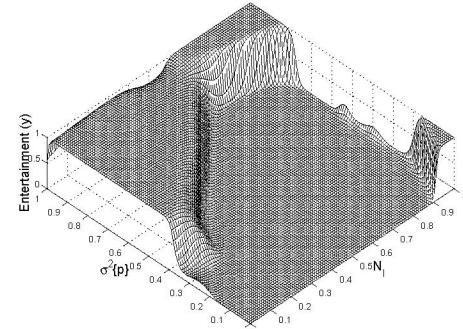
(a)  $E\{r_t\} = 1.0$  (Slow),  $H = 0.33$  (Low)



(b)  $E\{r_t\} = 1.0$  (Slow),  $H = 0.66$  (Average)



(c)  $E\{r_t\} = 1.0$  (Slow),  $H = 1.0$  (High)



(d)  $E\{r_t\} = 0.0$  (Fast),  $H = 0.33$  (Low)

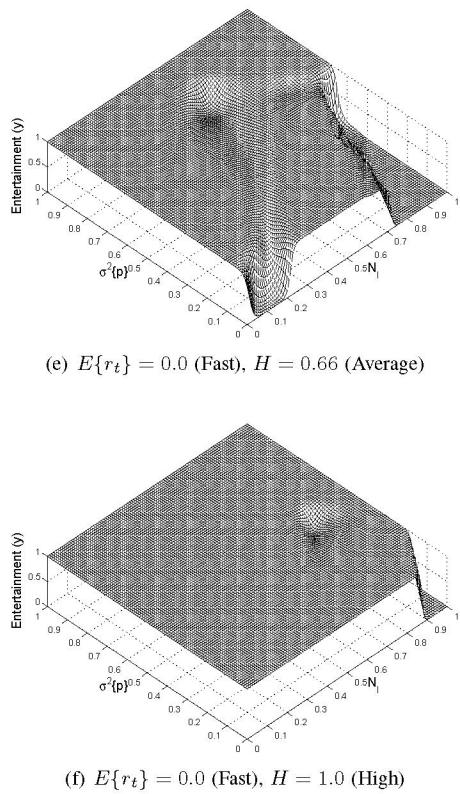


Fig. 2. The trained ANN ( $f = 95.85$ ) that yields the highest validation performance (90.00%): ANN output  $y$  (entertainment value) with regards to  $\sigma^2\{p\}$  and  $N_I$  for all six combinations of two  $E\{r_t\}$  states (Slow, Fast) and three  $H$  states (Low, Average, High).

is low and  $\sigma^2\{p\}$  value is high or when their  $N_I$  value is high and  $\sigma^2\{p\}$  value is low (see Fig. 2(b)). Finally, high curiosity is rarely preferred by slow children and this occurs only when their  $N_I$  values are low independently of their  $\sigma^2\{p\}$  value (Fig. 2(c)).

The obtained effects of curiosity in reported entertainment are consistent, in part, with previous studies on the Bug-Smasher game [6]. In that study the relation between challenge, curiosity and average response time was reported through a lower scale experiment of 28 children. It was found that fast children liked games independently of curiosity whereas children reacting slowly with the playground preferred games of low curiosity levels.

## VII. CONCLUSIONS & DISCUSSION

This paper introduced feature selection methods for obtaining minimal feature subsets that successfully model children's notion of entertainment through the Bug-Smasher game played on the Playware playground. More specifically, the nBest and the SFS feature selection methods were applied and compared demonstrating the ability of SFS in finding feature subsets that yield higher validation performance.

The fittest ANN solution presented derives from a feature subset of four features:  $\{E\{r_t\}, \sigma^2\{p\}, N_I, H\}$ . Experi-

ments with additional features (inputs of the ANN) could not improve the model's validation performance. This model manages to map between children's average response time, the variance of their force pressure on the tiles, the number of interactions with the playground, the game feature of curiosity and the children's notion of gameplay entertainment with a cross-validation accuracy of 77.77% (binomial-distributed p-value = 0.0019). The main reason for not obtaining a higher cross-validation performance appears to be the experimental noise existent in the self-reports designed for comparative entertainment (fun) analysis. Moreover, the learned mapping between  $\{E\{r_t\}, \sigma^2\{p\}, N_I, H\}$  and children's notion of entertainment showed that, in general, fast responding children show a preference for high curiosity games whereas slow responding children tend to prefer games of low curiosity. The obtained results are consistent with previous work on the impact of the factors of challenge and curiosity and the average response time in Playware games [6].

The main limitation of the proposed approach lies within the complexity of entertainment as a mental state. The generated  $y$  value cannot be regarded as a mental affective state approximator but as a correlate of expressed children's preferences on entertainment. However, this correlate serves the purposes of this work well as far as entertainment modeling is concerned. In addition, Malone's entertainment factor of fantasy is omitted from the results in this paper since the focus is on the contribution of the opponent behaviors to the generation of entertainment; however, fantasy's positive impact on reported entertainment has been reported in a previous study [28].

Even though the comparative fun protocol (2-AFC) used serves well the purpose of this work, a 4-alternative forced choice (4-AFC) approach is considered for future experiment protocol design. Children will be able to choose among the following alternatives: one game is more "fun" than the other (2-AFC), both games are equally "fun", neither game was "fun". This protocol would provide more information for the machine learning process and eliminate the noise generated by 2-AFC.

The entertainment modeling approach presented here demonstrates generality over the majority of action games created with Playware since the quantitative measures of challenge and curiosity are estimated through the generic features of speed and spatial diversity of the opponent on the game's surface. Thus, these or similar measures could be used to adjust player satisfaction in any future game development on the Playware tiles. However, each game demonstrates individual entertainment features that might need to be extracted and added on the proposed measures and therefore, more games of the same and/or other genres need to be tested to cross-validate this hypothesis.

The proposed approach can be used for adaptation of the game opponents (e.g. bugs) according to the player's individual playing style, based on reaction time, recorded pressure on tiles and amount of interactions, and as far as the curiosity factor of entertainment is concerned. Given the

real-time average response time of a child, the variance of his/her pressure forces on the tiles and the number of times he/she interacts with the environment, the partial derivative of the model output  $\partial y / \partial H$  can be used to appropriately adjust the level of entropy (curiosity) of the opponent for the entertainment value  $y$  to be augmented. Such a direction constitutes an example of future work on Playware and computer games. The level of engagement or motivation of the user/player/gamer of such interactive environments may be identified and augmented by the use of the presented approaches.

#### ACKNOWLEDGMENTS

The authors would like to thank Henrik Jørgensen and all children of Henriette Hørlücks and Rosengårdskolen Schools, Odense, Denmark that participated in the experiments.

The tiles were designed by C. Isaksen from Isaksen Design and parts of their hardware and software implementation were collectively done by A. Derakhshan, F. Hammer, T. Klitbo and J. Nielsen. KOMPAN, Mads Clausen Institute, and Danfoss Universe also participated in the development of the tiles.

This work was in part supported by the Danish National Research Council (project no: 274-05-0511).

#### REFERENCES

- [1] G. N. Yannakakis and J. Hallam, "A scheme for creating digital entertainment with substance," in *Proceedings of the Workshop on Reasoning, Representation, and Learning in Computer Games, 19th International Joint Conference on Artificial Intelligence (IJCAI)*, August 2005, pp. 119–124.
- [2] H. H. Lund, T. Klitbo, and C. Jessen, "Playware technology for physically activating play," *Artifical Life and Robotics Journal*, vol. 9, no. 4, pp. 165–174, 2005.
- [3] T. W. Malone, "What makes computer games fun?" *Byte*, vol. 6, pp. 258–277, 1981.
- [4] N. Postman, *The Disappearance of Childhood*. London: Allen, 1983.
- [5] S. Kline, *Out of the Garden: Toys and Children's Culture in the Age of Marketing*. Verso, 1993.
- [6] G. N. Yannakakis, H. H. Lund, and J. Hallam, "Modeling Children's Entertainment in the Playware Playground," in *Proceedings of the IEEE Symposium on Computational Intelligence and Games*. Reno, USA: IEEE, May 2006, pp. 134–141.
- [7] P. Devijver and J. Kittler, *Pattern Recognition - A Statistical Approach*. Engelwood cliffs, NJ: Prentice-Hall, 1982.
- [8] M. Csikszentmihalyi, *Flow: The Psychology of Optimal Experience*. New York: Harper & Row, 1990.
- [9] P. Sweetser and P. Wyeth, "GameFlow: A Model for Evaluating Player Enjoyment in Games," *ACM Computers in Entertainment*, vol. 3, no. 3, July 2005.
- [10] N. Lazzaro, "Why we play games: Four keys to more emotion without story," XEO Design Inc., Technical Report, 2004.
- [11] R. Koster, *A Theory of Fun for Game Design*. Paraglyph Press, 2005.
- [12] J. Read, S. MacFarlane, and C. Cassey, "Endurability, engagement and expectations," in *Proceedings of International Conference for Interaction Design and Children*, 2002.
- [13] P. Vorderer, T. Hartmann, and C. Klimmt, "Explaining the enjoyment of playing video games: the role of competition," in *ICEC conference proceedings 2003: Essays on the future of interactive entertainment*, D. Marinelli, Ed. Pittsburgh: Carnegie Mellon University Press, pp. 107–120.
- [14] D. Choi, H. Kim, and J. Kim, "Toward the construction of fun computer games: Differences in the views of developers and players," *Personal Technologies*, vol. 3, no. 3, pp. 92–104, September 1999.
- [15] H. Iida, N. Takeshita, and J. Yoshimura, "A metric for entertainment of boardgames: its implication for evolution of chess variants," in *IWEC2002 Proceedings*, R. Nakatsu and J. Hoshino, Eds. Kluwer, 2003, pp. 65–72.
- [16] G. N. Yannakakis and J. Hallam, "Evolving Opponents for Interesting Interactive Computer Games," in *From Animals to Animats 8: Proceedings of the 8th International Conference on Simulation of Adaptive Behavior (SAB-04)*, S. Schaal, A. Ijspeert, A. Billard, S. Vijayakumar, J. Hallam, and J.-A. Meyer, Eds. Santa Monica, LA, CA: The MIT Press, July 2004, pp. 499–508.
- [17] ———, "A Generic Approach for Obtaining Higher Entertainment in Predator/Prey Computer Games," *Journal of Game Development*, vol. 1, no. 3, pp. 23–50, December 2005.
- [18] G. N. Yannakakis, "AI in Computer Games: Generating Interesting Interactive Opponents by the use of Evolutionary Computation," Ph.D. thesis, University of Edinburgh, November 2005.
- [19] G. N. Yannakakis and J. Hallam, "Towards Capturing and Enhancing Entertainment in Computer Games," in *Proceedings of the 4th Hellenic Conference on Artificial Intelligence, Lecture Notes in Artificial Intelligence*, vol. 3955. Heraklion, Greece: Springer-Verlag, May 2006, pp. 432–442.
- [20] G. N. Yannakakis, J. Hallam, and H. H. Lund, "Capturing Entertainment through Heart-rate Dynamics in the Playware Playground," in *Proceedings of the 5th International Conference on Entertainment Computing, Lecture Notes in Computer Science*, vol. 4161. Cambridge, UK: Springer-Verlag, 2006, pp. 314–317.
- [21] P. Rani, N. Sarkar, and C. Liu, "Maintaining optimal challenge in computer games through real-time physiological feedback," in *Proceedings of the 11th International Conference on Human Computer Interaction*, 2005.
- [22] S. McQuiggan, S. Lee, and J. Lester, "Predicting User Physiological Response for Interactive Environments: An Inductive Approach," in *Proceedings of the 2nd Artificial Intelligence for Interactive Digital Entertainment Conference*, 2006, pp. 60–65.
- [23] G. Andrade, G. Ramalho, H. Santana, and V. Corruble, "Extending reinforcement learning to provide dynamic game balancing," in *Proceedings of the Workshop on Reasoning, Representation, and Learning in Computer Games, 19th International Joint Conference on Artificial Intelligence (IJCAI)*, August 2005, pp. 7–12.
- [24] M. A. Verma and P. W. McOwan, "An adaptive methodology for synthesising Mobile Phone Games using Genetic Algorithms," in *Congress on Evolutionary Computation (CEC-05)*, Edinburgh, UK, September 2005, pp. 528–535.
- [25] R. Hunnicke and V. Chapman, "AI for Dynamic Difficulty Adjustment in Games," in *Proceedings of the Challenges in Game AI Workshop, 19th Nineteenth National Conference on Artificial Intelligence (AAAI'04)*, 2004.
- [26] P. Spronck, I. Sprinkhuizen-Kuyper, and E. Postma, "Difficulty Scaling of Game AI," in *Proceedings of the 5th International Conference on Intelligent Games and Simulation (GAME-ON 2004)*, 2004, pp. 33–37.
- [27] R. Pfeifer and C. Scheier, *Understanding Intelligence*. Cambridge, MIT Press, 1999.
- [28] G. N. Yannakakis, J. Hallam, and H. H. Lund, "Comparative Fun Analysis in the Innovative Playware Game Platform," in *Proceedings of the 1st World Conference for Fun 'n Games*, 2006, pp. 64–70.
- [29] R. L. Mandryk, K. M. Inkpen, and T. W. Calvert, "Using Psychophysiological Techniques to Measure User Experience with Entertainment Technologies," *Behaviour and Information Technology (Special Issue on User Experience)*, vol. 25, no. 2, pp. 141–158, 2006.
- [30] C. Beal, J. Beck, D. Westbrook, M. Atkin, and P. Cohen, "Intelligent modelling of the user in interactive entertainment," in *Proceedings of the AAAI Spring Symposium on Artificial Intelligence and Interactive Entertainment*, Stanford, 2002, pp. 8–12.
- [31] X. Yao, "Evolving artificial neural networks," in *Proceedings of the IEEE*, vol. 87, no. 9, 1999, pp. 1423–1447.
- [32] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: University of Michigan Press, 1975.
- [33] D. J. Montana and L. D. Davis, "Training feedforward neural networks using genetic algorithms," in *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89)*. San Mateo, CA: Morgan Kaufmann, 1989, pp. 762–767.
- [34] E. Haapalainen, P. Laurinen, P. Junno, H. Tuovinen, and J. Roening, "Methods for classifying spot welding process: A comparative study of performance," in *IEA/AIE*, 2005, pp. 412–421.

# Modeling Player Experience in Super Mario Bros

Chris Pedersen, Julian Togelius and Georgios N. Yannakakis

**Abstract**— This paper investigates the relationship between level design parameters of platform games, individual playing characteristics and player experience. The investigated design parameters relate to the placement and sizes of gaps in the level and the existence of direction changes; components of player experience include fun, frustration and challenge. A neural network model that maps between level design parameters, playing behavior characteristics and player reported emotions is trained using evolutionary preference learning and data from 480 platform game sessions. Results show that challenge and frustration can be predicted with a high accuracy (77.77% and 88.66% respectively) via a simple single-neuron model whereas model accuracy for fun (69.18%) suggests the use of more complex non-linear approximators for this emotion. The paper concludes with a discussion on how the obtained models can be utilized to automatically generate game levels which will enhance player experience.

**Keywords:** Platform games, player satisfaction, content creation, fun, preference learning, entertainment modeling, neuroevolution

## I. INTRODUCTION

Numerous theories exist regarding what makes computer games fun, as well as which aspects contribute to other types of player experience such as challenge, frustration and immersion [1], [2], [3], [4], [5]. These theories have originated in different research fields and in many cases independently of each other (however, there is substantial agreement on several counts, e.g. regarding the importance of challenge and learnability for making a game fun). While useful high-level guidance for game design, none of these theories is quantitative — derived models of player experience are not mathematical expressions — and they tend to apply to games in general rather than specific aspects of specific games. This means that if we want to develop algorithms that design or adapt games (or aspects of games) automatically, we have to make several auxiliary assumptions in order to achieve the necessary specificity and preciseness of our models.

It seems clear that we need empirical research on particular games to acquire such models. Recently, research in player satisfaction modeling has focused on empirically measuring the effects on player experience of changing various aspects of computer games, such as non-player character (NPC) playing styles in the Pac-Man game [6]. Similarly, efficient quantitative models of player satisfaction have been constructed using in-game data, questionnaires and physiological measurements in augmented-reality games [7].

At the same time, a parallel research direction aims to find methods for automatically generating entertaining game

content. Automatic content generation is likely to be of great importance to computer game development in the future; both offline, for making the game development process more efficient (design of content such as environments and animations now consume a major part of the development budget for most commercial games) and online, for enabling new types of games based on player-adapted content. These efforts see some aspect of a game as variable, define a fitness (“goodness”) function based on a theory of player satisfaction, and use a learning or optimization algorithm to change the variable aspect of the game so as to become more “fun” according to some definition. The literature on this is so far scarce, as it is a new research direction. The aspects of games that have been optimized include:

- Environments, such as tracks for racing games [8] and levels for platform games [9].
- Narrative [10]
- Rules for board games [11], [12] and Pac-Man-like games [13].

What the above studies have in common is that the fitness or cost functions used for optimization have been somewhat arbitrary, in that they have been based on intuition in combination with some qualitative theory of player experience. Optimization of game aspects based on empirically derived models have so far been limited to the impact of NPC behavior [6] and the adjustment of NPC behavioral parameters for maximizing satisfaction in games [14]. To the best of our knowledge, game content such as rules or environments has not been generated based on empirically derived models. We consider better modeling of player experience to be of utmost importance for making automatic content generation techniques more sophisticated and usable.

The work we describe in this paper is concerned with the construction of computational models of player experience, derived from gameplay interaction, which can be used as fitness functions for game content generation. We use a modified version of a classic platform game for our experiments and collect player data through the Internet.

In the following, we describe the game used for our experiments; which player interaction data was collected and how; the preference learning method we used to construct player experience models; how feature selection was used to reduce the number of features used in the model; results of an initial statistical analysis; and the results of training nonlinear perceptrons to approximate the functions mapping between selected gameplay features and aspects of player experience. Finally, we discuss how the induced models will be used for automatically generating game content. This paper significantly extends [15] in which only the core ideas

The authors are with the Center for Computer Games Research, IT University of Copenhagen, Rued Langgaards Vej 7, DK-2300 Copenhagen S, Denmark. Emails: gammabyte@gmail.com, {juto, yannakakis}@itu.dk

of the methodology proposed are outlined.

## II. TESTBED PLATFORM GAME

The test-bed platform game used for our studies is a modified version of Markus Persson's *Infinite Mario Bros* (see Fig. 1) which is a public domain clone of Nintendo's classic platform game *Super Mario Bros*. The original Infinite Mario Bros is playable on the web, where Java source code is also available<sup>1</sup>.

The gameplay in Super Mario Bros consists of moving the player-controlled character, Mario, through two-dimensional levels, which are viewed sideways. Mario can walk and run to the right and left, jump, and (depending on which state he is in) shoot fireballs. Gravity acts on Mario, making it necessary to jump over holes (or gaps) to get past them. Mario can be in one of three states: Small (at the beginning of a game), Big (can crush some objects by jumping into them from below), and Fire (can shoot fireballs).

The main goal of each level is to get to the end of the level, which means traversing it from left to right. Auxiliary goals include collecting as many as possible of the coins that are scattered around the level, clearing the level as fast as possible, and collecting the highest score, which in part depends on the number of collected coins and killed enemies.

The presence of gaps and moving enemies are the main challenges of Mario. If Mario falls down a gap, he loses a life. If he touches an enemy, he gets hurt; this means losing a life if he is currently in the Small state, whereas if he is in the Big and Fire state he shifts to Small and Large state respectively. However, if he jumps so that he lands on the enemy from above, the outcome is dependent on the enemy: Most enemies (e.g. goombas, fireballs) die from this treatment; others (e.g. piranha plants) are not vulnerable to this and proceed to hurt Mario; finally, turtles withdraw into their shells if jumped on, and these shells can then be picked up by Mario and thrown at other enemies to kill them.

Certain items are scattered around the levels, either out in the open, or hidden inside blocks of brick and only appearing when Mario jumps at these blocks from below so that he smashes his head into them. Available items include coins which can be collected for score and for extra lives (every 100 coins), mushrooms which make Mario grow Big if he is currently Small, and flowers which make Mario turn into the Fire state if he is already Big.

No textual description can fully convey the gameplay of a particular game. Only some of the main rules and elements of Super Mario Bros are explained above; the original game is one of the world's best selling games, and still very playable more than two decades after its release in the mid-eighties. Its game design has been enormously influential and inspired countless other games, making it a good experiment platform for player experience modeling.

While implementing most features of Super Mario Bros, the standout feature of Infinite Mario Bros is the automatic generation of levels. Every time a new game is started,



Fig. 1. Test-bed game screenshot, showing Small Mario jumping over a piece of flat terrain surrounded by two gaps.

levels are randomly generated by traversing a fixed width and adding features (such as blocks, gaps and opponents) according to certain heuristics. In our modified version of Infinite Mario Bros most of the randomized placement of level features is fixed since we concentrate on a few selected game level parameters that affect game experience.

## III. DATA COLLECTION

Before any modeling could take place we needed to collect data to train the model on. We collected three types of data from hundreds of players over the Internet:

- 1) Controllable features of the game, i.e. the parameters used for level generation, and affecting the type and difficulty of the level. These were varied systematically to make sure all variants of the game were compared.
- 2) Gameplay characteristics, i.e. how the user plays the game. We measured statistical features such as how often and when the player jumped, ran, died etc. These features cannot be directly controlled by the game as they depend solely on the player's skill and playing style in a particular game level.
- 3) The player's experience of playing the game, measured through a 4-alternative forced choice questionnaire administered after playing two pairs of games with different controllable features and asking the players to rank the games in order of emotional preference.

Below we describe in detail which features were collected for each type of data.

### A. Controllable features

We modified the existing level generator to create levels according to four controllable parameters presented below. Three of these parameters are dedicated to the number, width and placement of gaps. The fourth parameter turns a new function, the direction switch, on or off.

- The number of gaps in the level,  $G$ .
- The average width of gaps,  $E\{G_w\}$ .
- The spatial diversity of gaps which is measured by the entropy of the number of gaps appearing in a number of

<sup>1</sup><http://www.mojang.com/notch/mario/>

$G$  (equally-spaced) segments of the level. The entropy of gap-placements  $H_g$  in the  $G$  segments is calculated and normalized into  $[0, 1]$  via (1):

$$H_g = \left[ -\frac{1}{\log G} \sum_{i=1}^G \frac{g_i}{G} \log \left( \frac{g_i}{G} \right) \right] \quad (1)$$

where  $g_i$  is the number of gap-placements into level segment  $i$ . If the gaps are placed in all  $G$  level segments uniformly then  $g_i = 1$  for all  $G$  parts and  $H_g$  will be 1; if all gaps are placed in one level segment  $H_g$  is zero. This controllable feature provides a notion of unpredictability of gaps and, therefore, jumps in the level. Unpredictability has been identified as an important factor of playing experience [16].

- Number of direction switches,  $S$ . No direction switch means that the player needs to move from left to right in order to complete the level, as in the original Super Mario Bros. If one or more direction switch is present, the level direction will be mirrored at certain points, forcing the player to turn around and go the other way, until reaching the end of the level or the next switch.

The selection of these particular controllable features was done after consulting game design experts, and with the intent to find features which were common to most, if not all, platform games.

Two states (low and high) for each of the four controllable parameters above are investigated. The combinations of these states result in  $2^4 = 16$  different variants of the game which are used in the user study designed. In the Super Mario Bros game investigated here the number of coins, opponents, coin blocks, powerup blocks and empty blocks are fixed to 15, 3, 4, 2, and 8 respectively.

### B. Gameplay features

Several statistical features are extracted from playing data which are logged during gameplay and include game completion time, time spent on various tasks (e.g. jumping, running), information on collected items (e.g. type and amount), killed enemies (e.g. type, amount, way of killing) and information on how the player died. The choice of those specific statistical features is made in order to cover a decent amount of Super Mario Bros playing behavior dynamics. In addition to the four controllable game features that are used to generate Super Mario Bros levels presented earlier, the following statistical features are extracted from the gameplay data collected and are classified in five categories: jump, time, item, death, kill and misc.

Jump: difference between the total number of jumps,  $J$ , minus the number of gaps,  $G$ ; number of jumps over gaps or without any purpose (e.g. to collect an item, to kill an opponent),  $J'$ ; difference between  $J'$  and the number of gaps,  $G$ ; and a jump difficulty heuristic,  $J_d$ , which is proportional to the number of Super Mario deaths due to gaps, number of gaps and average gap width.

Time: completion time,  $t_c$ ; playing duration of last life over total time spent on the level,  $t_{ll}$ ; percentage of time that the player: is standing still,  $t_s$ , running,  $t_r$ , is on large Mario mode,  $t_l$ , is on fire Mario mode,  $t_f$ , is on powerup mode,  $t_p$ , is moving left,  $t_L$ , is moving right,  $t_R$ , and is jumping,  $t_j$ .

Item: number of collected items (coins, destroyed blocks and powerups) over total items existent in the level,  $n_I$ ; number of times the player kicked an opponent shell,  $n_s$ ; number of coins collected over the total number of coins existent in the level,  $n_c$ ; number of empty blocks destroyed over the total number of empty blocks existent in the level,  $n_{eb}$ ; number of coin blocks pressed over the total number of coin blocks existent in the level,  $n_{cb}$ ; number of powerup blocks pressed over the total number of powerup blocks existent in the level,  $n_p$ ; and the sum of all blocks pressed or destroyed over the total number of blocks existent in the level  $n_b = n_{eb} + n_{cb} + n_p$ .

Death: number of times the player was killed: by an opponent,  $d_o$ ; by jumping into a gap,  $d_g$ ; by jumping into a gap over the total number of deaths  $d_j$ .

Kill: number of opponents died from stomping over the total number of kills,  $k_s$ ; number of opponents died from fire-shots over the total number of kills,  $k_f$ ; total number of kills over total number of opponents,  $k_T$ ; number of opponent kills minus number of deaths caused by opponents,  $k_P$ ; and number of cannonballs killed,  $k_c$ .

Misc: number of times the player shifted the mode (Small, Big, Fire),  $n_m$ ; number of times the run button was pressed,  $n_r$ ; number of ducks,  $n_d$ ; number of cannonballs spawned,  $n_{cs}$ ; and whether the level was completed or not  $C$  (boolean).

### C. Reported player experience and experimental protocol

We designed a game survey study to solicit pairwise emotional preferences of subjects playing different variants of the test-bed game by following the experimental protocol proposed in [7]. Each subject plays a predefined set of four games in pairs: a game pair of game  $A$  and game  $B$  played in both orders. The games played differ in the levels of one or more of the four controllable features presented previously. For each completed pair of games  $A$  and  $B$ , subjects report their emotional preference using a 4-alternative forced choice (4-AFC) protocol:

- game  $A$  [ $B$ ] was/felt more  $E$  than game  $B$  [ $A$ ] game (cf. 2-alternative forced choice);
- both games were/felt equally  $E$  or
- neither of the two games was/felt  $E$ .

Where  $E$  is the emotional state under investigation and contains *fun, challenging, boring, frustrating, predictable* and *anxious*. The selection of these six states is based on their relevance to computer game playing and their popularity when it comes to game-related user studies [17]. In this initial investigation of player experience we focus only on three emotions: *fun, challenge* and *frustration*.

Data is collected over the Internet. Users are recruited via posts on blogs and mailing lists and directed to a web page containing a Java applet implementing the game and questionnaire<sup>2</sup>. As soon as the four games are played and the questionnaire is completed, all the features (controllable, gameplay and player experience) are saved in a database at the server hosting the website and applet. Data collection is still in progress and at the moment of writing, 181 subjects have participated in the survey experiment. The minimum number of experiment participants required is determined by  $C_2^{16} = 120$ , this being the number of all combinations of 2 out of 16 game variants. The experimental protocol is designed in such a way that at least 2 preference instances should be obtained for each pair of the 16 game variants played in both orders (1 preference instance per playing order). The analysis presented in this paper is based on the 240 game pairs (480 game sessions) played by the first 120 subject participants.

#### IV. PREFERENCE LEARNING FOR MODELING PLAYER EXPERIENCE

Based on the data collected in the process described above, we try to approximate the function from gameplay features (e.g. number of coins gathered) and controllable game level features (e.g. number of gaps) to reported emotional preferences using neuroevolutionary preference learning.

The data is assumed to be a very noisy representation of the underlying function, given the high level of subjectivity of human preferences and the expected variation in playing styles. Together with the limited amount of training data, this makes overfitting a potential hazard and mandates that we use a robust function approximator. We believe that a non-linear function such as an artificial neural network (ANN) is a good choice for approximating the mapping between reported emotions and input data. Thus, a simple single-neuron (perceptron) is utilized for learning the relation between features (ANN inputs) — selected from feature selection schemes presented in Section V — and the value of the investigated emotional preference (ANN output) of a game. The main motivation for using a single neuron instead of a multi-layered perceptron (MLP) in this study is that we want to be able to analyze the trained function approximator. While an MLP can potentially approximate the function investigated with a higher accuracy, it is much easier for a human to understand the obtained function when represented as a single-neuron ANN.

The single neuron uses the sigmoid (logistic) activation function; connection weights take values from -5 to 5 to match the normalized input values that lie in the [0, 1] interval. Since there are no prescribed target outputs for the learning problem (i.e. no differentiable output error function), ANN training algorithms such as back-propagation are inapplicable. Learning is achieved through artificial evolution by following the preference learning approach presented in

<sup>2</sup>The game and questionnaire are available at [www.bluenight.dk/mario.php](http://www.bluenight.dk/mario.php)

[18]. A generational genetic algorithm (GA) is implemented, using a fitness function that measures the difference between the subject's reported emotional preferences and the relative magnitude of the corresponding model (ANN) output.

#### V. FEATURE SELECTION

We would like our model to be dependent on as few features as possible, both to make it easier to analyze, and to make it more useful for incorporation into future games for purposes of e.g. content creation. Therefore, feature selection is utilized to find the feature subset that yields that most accurate user model and save computational effort of exhaustive search on all possible feature combinations. The quality of the predictive model constructed by the preference learning outlined above depends critically on the set of input data features chosen. Using the extracted features described earlier the *n best individual feature selection* (nBest), the *Sequential Forward Selection* (SFS) and the *Perceptron Feature Selection* (PFS) schemes are applied and compared.

##### A. nBest

nBest feature selection ranks the features used individually in order of model performance; the chosen feature set of size n is then the first n features in this ranking. The nBest method is used for comparative purposes, being the most popular technique for feature selection.

##### B. SFS

SFS is a bottom-up search procedure where one feature is added at a time to the current feature set. The feature to be added is selected from the subset of the remaining features such that the new feature set generates the maximum value of the performance function over all candidate features for addition. The SFS method has been successfully applied to a wide variety of feature selection problems, yielding high performance values with minimal feature subsets [7], [19]

##### C. PFS

The third method we investigate is Rosenblatt's perceptron as a methodology for selecting appropriate feature subsets. Our algorithm which is similar to [20] is adjusted to match preference learning problems. Thus, the perceptron used employs the sigmoid activation function in a single output neuron. The ANN's initial input vector has the size of the number of features examined. The perceptron feature selection (PFS) procedure is as follows:

**Step 1** Use artificial evolution to train the perceptron on the pairwise preferences (see Section IV). Performance of the perceptron is evaluated through 3-fold cross-validation. The initial input vector consists of all features extracted  $\mathcal{F}$  (40 in this paper).

**Step 2** Eliminate all features  $\mathcal{F}'$  whose corresponding absolute connection weight values are smaller than  $E\{|w|\} - \sigma\{|w|\}$ , where w is the connection weight vector.

**Step 3** If  $\mathcal{F}' = \emptyset$  continue to Step 4, otherwise use the remaining features and go to Step 1.

**Step 4** Evaluate all feature subsets obtained using the neuro-evolution preference learning approach presented in Section IV.

Note that all three methods are incomplete. Neither is guaranteed to find the optimal feature set since neither searches all possible combinations (they are all variants of hill-climbing). To evaluate the performance of each input feature subset, the available data is randomly divided into thirds and training and validation data sets consisting of 2/3 and 1/3 of the data respectively are assembled. The performance of each user model is measured through the average classification accuracy of the model in three independent runs using the 3-fold cross-validation technique on the three possible independent training and validation data sets. Since we are interested in the minimal feature subset that yields the highest performance we terminate the SFS selection procedure when an added feature yields equal or lower validation performance to the performance obtained without it. On the same basis, we store all feature subsets selected by PFS and explore the highest performing subset starting with the smallest feature subset generated.

## VI. STATISTICAL ANALYSIS

This section describes testing for correlations between playing order, controlled features and gameplay features and the reported emotions of fun, challenge and frustration.

To check whether the order of playing Super Mario game variants affects the user's judgement of emotional preferences, we follow the order testing procedure described in [6] which is based on the number of times that the subject prefers the first or the second game in both pairs. The statistical analysis shows that order of play does not affect the emotional preferences of fun and frustration; however a statistically significant effect is observed in challenge preferences ( $p$ -value = 0.006). The effect reveals a preference for the second game played which implies the existence of random noise in challenge preference expression. On the other hand, the insignificant order effects of fun and frustration, in part, demonstrate that effects such as a user's possible preference for the very first game played and the interplay between reported emotions and familiarity with the game are statistically insignificant.

More importantly, we performed an analysis for exploring statistically significant correlations between subject's expressed preferences and extracted features. Correlation coefficients are obtained through  $c(\mathbf{z}) = \sum_{i=1}^{N_s} \{z_i/N_s\}$ , where  $N_s$  is the total number of game pairs where subjects expressed a *clear preference* for one of the two games (e.g.  $A \succ B$  or  $A \prec B$ ) and  $z_i = 1$ , if the subject preferred the game with the larger value of the examined feature and  $z_i = -1$ , if the subject chooses the other game in the game pair  $i$ . Note that,  $N_s$  is 161, 189 and 151 respectively, for reported fun, challenge and frustration.

The variation of the  $N_s$  numbers above indicates, in part, the difficulty in expressing a clear emotional preference on different game variants. The percentage of  $A \succ B$

and  $A \prec B$  selection occurrences over all 240 preference instances for different emotional states varies from 78.7% (challenge) to 62.9% (frustration). These percentages provide some first evidence that the selected game level and rule parameters have an dissimilar impact on the emotional states investigated. For instance, challenge appears to be very much affected by varying the selected parameters whereas frustration, on the contrary, does not appear as an emotion which is directly affected by variations in the game.

TABLE I  
TOP TEN STATISTICALLY SIGNIFICANT ( $P$ -VALUE < 1%) CORRELATION COEFFICIENTS BETWEEN REPORTED EMOTIONS AND EXTRACTED FEATURES.

|          | Fun   |            | Challenge |          | Frustration |
|----------|-------|------------|-----------|----------|-------------|
| $n_s$    | 0.345 | $C$        | -0.600    | $C$      | -0.826      |
| $n_{cb}$ | 0.311 | $n_p$      | -0.480    | $n_p$    | -0.815      |
| $k_T$    | 0.256 | $d_j$      | 0.469     | $n_{cb}$ | -0.688      |
| $n_r$    | 0.253 | $d_g$      | 0.447     | $d_g$    | 0.578       |
| $t_L$    | 0.237 | $J_d$      | 0.439     | $d_j$    | 0.564       |
| $k_P$    | 0.222 | $E\{G_w\}$ | 0.409     | $n_I$    | -0.544      |
| $t_r$    | 0.192 | $n_d$      | -0.368    | $t_s$    | 0.520       |
|          |       | $t_{ll}$   | -0.312    | $k_f$    | -0.515      |
|          |       | $n_{cb}$   | -0.292    | $t_{ll}$ | -0.513      |
|          |       | $G$        | -0.287    | $n_c$    | -0.511      |

### A. Fun

Statistically significant correlations are observed between reported fun and seven features: number of times the player kicked a turtle shell, proportion of coin blocks that were "pressed" (jumped at from below), proportion of opponents that were killed, number of times the run button was pressed, proportion of time spent moving left, number of enemies killed minus times died, and proportion of time spent running. All of these were positive correlations.

Such correlations draw a picture of most players enjoying a fast paced game that includes near-constant progress, plenty of running, many enemies killed and many coins collected from bouncing off the coin blocks. One might argue that this picture fits with the concept of *Flow*, in that the player makes unhindered progress [3]. However, the Flow concept also includes a certain level of challenge, and no features that signify challenge are associated with fun in this case. It might be that players enjoy when the game is easy — at least when they only play a single level of the game.

The feature that correlates the most with fun preferences is kicking turtle shells. Kicking a turtle shell is a simple action which often results in the unfolding of a relatively complex sequence of events, as the shell might bounce off walls, kill enemies, fall into gaps etc. The fun inherent in setting of complex chains of events with simple actions is something many players can relate to and which features prominently in many games, but which is to our knowledge not part of any of the "established" theories of what makes games fun.

### B. Challenge

Eighteen features are significantly correlated with challenge. The ten most highly correlated are (+/- in parenthesis signifies positive or negative correlation): whether the level was completed (-), proportion of power-up blocks pressed (-), proportion of Mario deaths that were due to falling into a gap (+), number of times Mario died from falling into a gap (+), jump difficulty (+), average width of gaps (+), number of times Mario ducked (-), proportion of time spent in the last life (-), proportion of coin blocks that were pressed (-), and the number of gaps (-). In addition, a weaker but still significant positive correlation was found between gap entropy,  $H_g$ , and challenge.

A first observation is that it is obviously much easier to predict challenge than to predict fun — many more features are significantly correlated, and the correlations are stronger. It also seems that challenge is somehow orthogonal to fun, as almost none of the features that are correlated with challenge are correlated with fun. The exception is the proportion of coin blocks pressed, but while this feature is positively correlated with fun it is negatively correlated with challenge. (This is somewhat expected: if the level is so hard that the player has to struggle to survive it, she does not have time to make detours in order to collect more coins.)

Most of the correlations are easy to explain. That a level is perceived as less challenging if you complete it should not come as a surprise to anyone. Likewise, we can understand that players think a level is hard when they repeatedly die from falling into gaps. Three particularly interesting correlations are those between the controllable features and challenge: increase in gap width,  $E\{G_w\}$ , and gap entropy,  $H_g$ , imply increased challenge whereas increased number of gaps,  $G$ , implies a linear decrease of challenge. These effects suggest that challenge can be controlled to a degree by changing the number, width and distribution of gaps.

The correlation with number of ducks would have been easy to explain — if it was positive. The main reason for ducking in Super Mario Bros (at least in the tested levels) is to avoid cannonballs. To the authors, cannons are perceived as some of the most difficult elements on a level. However, players reported lower challenge on levels where they ducked many times. We have yet to find an explanation for this.

### C. Frustration

Twenty-eight features are significantly correlated with frustration, and some of the correlations are extremely strong. Of the top ten correlated features, most are also in the top ten list for features correlated with challenge, and correlated in the same way. The exceptions are proportion of collected items (-), time spent standing still (+), proportion of killed opponents that were killed with fireballs (-), and proportion of coins collected (-).

From these new features, it seems that a frustrated player is most likely one that spends time standing still and thinking about how to overcome the next obstacle; is far too busy overcoming obstacles to collect coins and power-ups; and

as a result of not collecting power-ups is rarely in the Fire Mario state (necessary to shoot fireballs). But frustration can also be very well predicted from not winning the level and from falling into gaps often, just like challenge.

### D. Controllable features and intra-correlations

When only looking at linear correlations, it would appear that fun is not connected to any of the four controllable features. Fun is also less strongly correlated with gameplay features than is the case for challenge and frustration. The latter two emotions are easier to model with linear models, and are also strongly correlated with controllable features, namely gap entropy, gap width and number of gaps.

The three emotions are also all significantly ( $p$ -value  $< 0.001$ ) correlated to each other. The correlation coefficient  $c(\mathbf{z})$  between challenge and fun and between challenge and frustration is 0.346 and 0.462, respectively, while the corresponding correlation between fun and frustration is -0.284. The positive effect of challenge on fun and frustration, combined with the negative effect of fun on frustration indicate the nonlinearity (and possibly complexity) of those emotions' interrelationships.

## VII. NONLINEAR PREFERENCE MODELLING

The correlations calculated above provide linear relationships between individual features and reported emotions. However, these relationships are most likely more complex than can be captured by linear models. The aim of the analysis presented below is to construct non-linear computational models for reported emotions and analyze the relationship between the selected features and expressed preferences.

For this purpose we evolve weights for nonlinear perceptrons as described in Section IV. The weights of the highest performing networks are presented in Table II. All evolved networks performed much better than networks with random weights, which reached chance level prediction accuracy.

TABLE II  
LEARNING FROM PREFERENCES: FEATURES AND CORRESPONDING CONNECTION WEIGHTS FOR HIGHEST PERFORMING ANNS

|       | Fun    | Challenge | Frustration |
|-------|--------|-----------|-------------|
| $t_L$ | 4.905  | $t_s$     | -1.703      |
| $k_s$ | 0.942  | $J_d$     | 3.805       |
| $L$   | -3.873 | $n_{eb}$  | -1.502      |
|       |        | $k_c$     | 1.073       |
|       |        | $k_s$     | -0.189      |
|       |        | $t_{ll}$  | -1.851      |
|       |        | $J_d$     | -0.995      |
|       |        | $d_g$     | 0.233       |

### A. Fun

In the comparison between the three different selection mechanisms applied it is evident that SFS has advantages over nBest and PFS for fun preferences (see Fig. 2(a)). nBest achieves a satisfactory performance (67.92%) but requires 10 features as inputs to the ANN. PFS generates the lowest classification accuracies; its best network has an accuracy of 63.52% with a selected subset of 11 input features.

The best obtained perceptron model of fun preferences is designed by SFS. This model achieves a performance

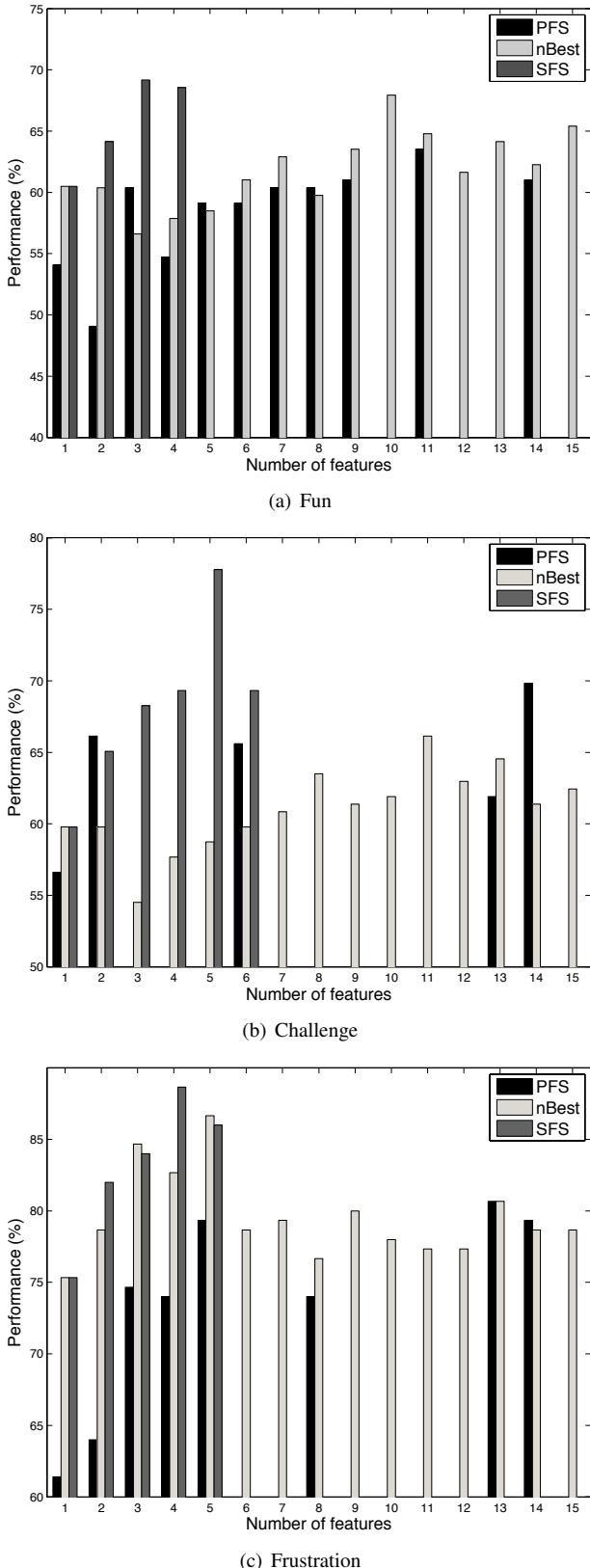


Fig. 2. Performance comparison of feature selection mechanisms on emotional preferences.

of 69.18% which is with a selected feature subset of size three. The selected perceptron input vector consists of the time spent moving left  $t_L$ , the number of opponents died from stomping over the total number of kills,  $k_s$ , and the controllable *switching* feature which is defined as the percentage of level played in the left direction  $L^3$ .

Fun is the least correlated of the three modeled emotions, and the hardest to model with a nonlinear perceptron as well. Still, it's remarkable that this complex emotion can be predicted to a moderate degree simply by observing that Mario keeps running left and kills enemies by stomping.

### B. Challenge

The best-performing ANN for challenge prediction has an accuracy of 77.77%. It is more complex than the best fun predictor, using five features: time spent standing still (-), jump difficulty (+), proportion of coin blocks pressed (-), number of cannonballs killed (-) and proportion of kills by stomping (-). While the jump difficulty heuristic has the largest corresponding weight — a testament to the central role of gap placement and size for challenge — it is also the only feature related to gaps used by this model, pointing to the adequateness of this particular heuristic.

### C. Frustration

Our best evolved ANN for predicting frustration has an accuracy of 88.66%. We can predict with near-certainty whether the player is frustrated by the current game by just calculating the time spent standing still (+), the proportion of time spent on last life (-), the jump difficulty (-), and the proportion of deaths due to falling in gaps (+).

Somewhat surprisingly, time spent standing still counts *against* challenge, whereas it is a strong *positive* predictor of frustration. This observation could be valuable if trying to design a feedback system that keeps the game challenging but not frustrating. Another feature that has different effect on challenge and frustration is jump difficulty, where frustration is connected with *lower* jump difficulty. Maybe the player gets frustrated by falling into gaps that she knows are not that hard.

That the player feels frustrated when dying after a short time during his last life is understandable — many players feel that their last attempt should be their best. Additionally, a high frustration level can cause the player to care less about the game and play worse in her final life.

## VIII. DISCUSSION

While we have found relatively good predictors for all three emotions, two problems remain: the predictions (at least for fun and challenge) are still not as good as we would like them to be, and we cannot reliably predict fun from controllable features. As controllable features (such as level

<sup>3</sup>The  $L$  feature is there to correct for the fact that when the level direction switches, Mario moves right rather than left to move forward, and so  $t_L$  is diminished. This points to an oversight on our part when designing the gameplay features: we should have measured the time spent moving *towards the end of the level* rather than moving left.

design parameters) are those that we can vary, and therefore those that can be optimized by evolution or other global optimizers, we need to be able to predict emotions at least partly from controllable features.

This points to the need for better models and/or features. First of all, we will try to induce multi-layer perceptron models of emotional preferences. MLPs have the advantage of universal approximation capacity; in particular, combinatorial relationships (such as XOR) can be represented. We might very well have a situation where one controllable feature (such as gap width) can be both negatively and positively connected with an emotion (such as frustration) depending on the player's playing style, as measured through gameplay features (such as number of jumps). Such relationships can be captured by MLPs but not by nonlinear perceptrons.

Data collection is continuing at the time of writing, and probably at the time of reading (the reader is welcome to contribute by visiting the project's web site), the new data will be used to improve the accuracy of our predictions.

Depending on the success of finding predictors partly dependent on controllable features, we might need to design new controllable features or revise the existing ones. New features might include the number and type of enemies, the existence of dead ends in the level (forcing backtracking) etc.

After good models have been learned, evolutionary algorithms will be used to optimize the level design parameters (relating to gaps and switches) for different objectives. We hope to, this way, be able to generate levels that tailor the playing experience according to the needs of the game design (e.g. a challenging level combined with a non-frustrating experience). The success of our optimization attempts will be validated with further user studies.

Another question concerns the generality of the results gathered here — do they apply to just the players and the particular game tested here, or do they have wider applicability? We venture that, as Super Mario Bros more or less defined the platform game genre, the results apply to some extent to all games of the same genre. Further, the population of experimental subjects is believed to be very diverse, but this needs to be verified. A possible critique is that the emotions reported are those that have been elicited after only a few minutes of play. It is possible that challenge or variety (gap entropy) would factor in more if play sessions were longer, so subjects would have had a chance of getting bored with the game.

## IX. CONCLUSIONS

We designed a user study focused on a version of the Super Mario Bros platform game, in which a population of subjects played in a number of different versions (mainly differing in the game environments encountered). Controllable features and emergent gameplay features were correlated with reported emotions during gameplay. We found a large number of statistically significant correlations, and were able to train good predictors of player emotions using preference learning and neuroevolution. These results will be improved upon and form the basis for attempts to automatically generate

environments for this game using artificial evolution with the induced player experience models as fitness functions.

## ACKNOWLEDGMENTS

The authors would like to thank Aki Järvinen and Markus Persson for insightful discussions, and all subjects that participated in the experiments.

## REFERENCES

- [1] C. Bateman and R. Boon, *21st Century Game Design*. Charles River Media, 2005.
- [2] K. Isbister and N. Schaffer, *Game Usability: Advancing the Player Experience*. Morgan Kaufman, 2008.
- [3] M. Csikszentmihalyi, *Flow: the Psychology of Optimal Experience*. Harper Collins, 1990.
- [4] R. Koster, *A theory of fun for game design*. Paraglyph press, 2005.
- [5] J. Juul, *Half-real*. MIT Press, 2005.
- [6] G. N. Yannakakis and J. Hallam, "Towards optimizing entertainment in computer games," *Applied Artificial Intelligence*, vol. 21, pp. 933–971, 2007.
- [7] ———, "Entertainment modeling through physiology in physical play," *International Journal of Human-Computer Studies*, vol. 66, pp. 741–755, 2008.
- [8] J. Togelius, R. De Nardi, and S. M. Lucas, "Towards automatic personalised content creation in racing games," in *Proceedings of the IEEE Symposium on Computational Intelligence and Games*, 2007.
- [9] K. Compton and M. Mateas, "Procedural level design for platform games," in *Proceedings of the Artificial Intelligence and Interactive Digital Entertainment International Conference (AIIDE)*, 2006.
- [10] M. J. Nelson, C. Ashmore, and M. Mateas, "Authoring an interactive narrative with declarative optimization-based drama management," in *Proceedings of the Artificial Intelligence and Interactive Digital Entertainment International Conference (AIIDE)*, 2006.
- [11] C. Browne, "Automatic generation and evaluation of recombination games," Ph.D. dissertation, Queensland University of Technology, Brisbane, Australia, 2008.
- [12] J. Marks and V. Horn, "Automatic design of balanced board games," in *Proceedings of the Artificial Intelligence and Interactive Digital Entertainment International Conference (AIIDE)*, 2007, pp. 25–30.
- [13] J. Togelius and J. Schmidhuber, "An experiment in automatic game design," in *Proceedings of the IEEE Symposium on Computational Intelligence and Games*, 2008.
- [14] G. N. Yannakakis and J. Hallam, "Real-time Game Adaptation for Optimizing Player Satisfaction," *IEEE Transactions on Computational Intelligence and AI in Games*, 2009, (to appear).
- [15] C. Pedersen, J. Togelius, and G. Yannakakis, "Optimization of platform game levels for player experience," in *Proceedings of the Artificial Intelligence and Interactive Digital Entertainment International Conference (AIIDE)*, 2009.
- [16] T. W. Malone, "What makes computer games fun?" *Byte*, vol. 6, pp. 258–277, 1981.
- [17] R. L. Mandryk and M. S. Atkins, "A Fuzzy Physiological Approach for Continuously Modeling Emotion During Interaction with Play Environments," *International Journal of Human-Computer Studies*, vol. 65, pp. 329–347, 2007.
- [18] G. N. Yannakakis and J. Hallam, "Game and Player Feature Selection for Entertainment Capture," in *Proceedings of the IEEE Symposium on Computational Intelligence and Games*. Hawaii, USA: IEEE, April 2007, pp. 244–251.
- [19] G. N. Yannakakis, M. Maragoudakis, and J. Hallam, "Preference Learning for Cognitive Modeling: A Case Study on Entertainment Preferences," *IEEE Systems, Man and Cybernetics; Part A: Systems and Humans*, 2009, (to appear).
- [20] M. Mejia-Lavalle and G. Arroyo-Figueroa, "Power System Database Feature Selection Using a Relaxed Perceptron Paradigm," in *Proceedings of 5th Mexican International Conference on Artificial Intelligence, LNCS*. Springer Berlin/Heidelberg, 2006, pp. 522–531.

## TOWARDS OPTIMIZING ENTERTAINMENT IN COMPUTER GAMES

**Georgios N. Yannakakis** □ *Mærsk Mc-Kinney Møller Institute, University of Southern Denmark, Odense M, Denmark*

**John Hallam** □ *Mærsk Mc-Kinney Møller Institute, University of Southern Denmark, Odense M, Denmark*

□ *Mainly motivated by the current lack of a qualitative and quantitative entertainment formulation of computer games and the procedures to generate it, this article covers the following issues: It presents the features—extracted primarily from the opponent behavior—that make a predator/prey game appealing; provides the qualitative and quantitative means for measuring player entertainment in real time, and introduces a successful methodology for obtaining games of high satisfaction. This methodology is based on online (during play) learning opponents who demonstrate cooperative action. By testing the game against humans, we confirm our hypothesis that the proposed entertainment measure is consistent with the judgment of human players. As far as learning in real time against human players is concerned, results suggest that longer games are required for humans to notice some sort of change in their entertainment.*

Intelligent interactive opponents can provide more enjoyment to a vast gaming community of constant demand for more realistic, challenging, and meaningful entertainment (Fogel et al. 2004; Champandard 2004). However, given the current state-of-the-art in artificial intelligence (AI) in computer games, it is unclear which features of any game contribute to the satisfaction of its players, and thus it is also uncertain how to develop enjoyable games. Because of this lack of knowledge, most commercial and academic research in this area is fundamentally incomplete. The challenges we consider in this article are to provide qualitative and quantitative means for distinguishing a game's enjoyment value and to develop efficient AI tools to automatically generate entertainment for the player.

In our previous work (Yannakakis and Hallam 2004), we defined criteria that contribute to the satisfaction for the player, which map to

The authors would like to thank the anonymous reviewers for their valuable feedback and the players for their vital contribution to this work.

Address correspondence to G. N. Yannakakis, Mærsk Mc-Kinney Møller Institute, University of Southern Denmark, Odense M, Denmark. E-mail: georgios@mmti.sdu.dk

characteristics of the opponent behavior in computer games. According to our hypothesis, the player-opponent interaction—rather than the audiovisual features, the context, or the genre of the game—is the property that primarily contributes the majority of the quality features of entertainment in a computer game. Based on this fundamental assumption, we introduced a metric for measuring the real-time entertainment value of predator/prey games.

According to our second hypothesis, entertainment is generated when adaptive learning procedures occur in real time. This allows for opponents to learn while playing against the player and adapt with regards to his/her strategy. When such a mechanism is built upon cooperative opponents, it is more likely that the game's interest value improves drastically. Using the Pac-Man game as a test-bed (Yannakakis and Hallam 2004a, 2005) and focusing on the non-player characters' (NPC's) cooperative behavior, a robust online (i.e., while the game is played) neuro-evolution (i.e., automatic shaping of artificial neural networks by the use of artificial evolution) learning mechanism was presented, which was capable of increasing the game's interest as well as keeping that interest at high levels while the game was being played. This mechanism demonstrated high robustness and adaptability to changing hand-crafted player strategies in a relatively simple playing stage. Additional experiments in a predator/prey game of a more abstract design called Dead End (Yannakakis and Hallam 2004b) displayed the effectiveness of the proposed methodology in dissimilar games of the same genre and expanded the applicability of the method.

In the work presented here, apart from experiments with computer-guided players, human players are used for testing the Pac-Man game in a more complex stage. The main objective of this work is to establish the interest measure proposed as an efficient generic predator/prey game metric, by the approval of human judgment. In other words, we attempt to cross-correlate the human notion of interest to the proposed interest metric in one of the most representative test-beds of this computer games domain. Furthermore, we examine the online learning algorithm's abilities against humans and attempt to discover whether it maintains its robustness and adaptability under real conditions, that is, against human players.

## **LEARNING IN GAMES**

The majority of research on learning in games is built on board or card games. In the last decade, many researchers have been involved in the development of intelligent opponents in board or card games. Some of the attempts include evolutionary learning approaches applied, from tic-tac-toe (Fogel 1993), to checkers (Fogel [2002] among others) and Go (Richards et al. 1998). In Tesauro (2002), a temporal difference learning

mechanism generates computer opponents capable of beating even expert humans in backgammon. These games are board games simulated in computers and therefore sometimes people refer to them as “computer games.” However, when we refer to computer games, we refer to the category of commercial games played by NPCs in virtual worlds.

Based on the success of this mentioned research on board games, increasing computing power and the commercial possibilities of computer games, very recently, researchers have attempted to introduce AI into computer games and have discussed the theoretical perspective of learning in different categories of games. Laird (2002) surveys the state of research in using AI techniques in interactive computer games. He also provides a taxonomy of games and the importance of computer games as experimental environments for strong AI application. Furthermore, Isla and Blumberg (2002) suggest potential research directions in AI game development, emphasizing to the emotional state and the perceived information of the character. Taylor (2000) attempts to bridge the gap between game development and modern AI by proposing artificial life techniques for generating physically modeled characters.

Game AI researchers, in their majority, focus on the genre of first-person shooter (FPS) and real-time strategy (RTS) games, primarily because of their popularity and secondarily because of their open-source game engines. Alex J. Champandard (2004) is using an FPS game to propose and apply a plethora of forms of AI techniques (varying from simple scripting to adaptive learning) for specific tasks like movement, shooting, and weapon selection. Khoo (2002) developed an inexpensive AI technique based on the well-known Eliza program (Weizenbaum 1966), so that users get the impression of playing against humans instead of bots. In Cole et al. (2004), the parameters of the *Counter-Strike* built-in weapon selection rules are tuned by using artificial evolution. Furthermore, there have been attempts to mimic human behavior offline, from samples of human playing, in a specific virtual environment. Alternatively, dynamic scripting and evolutionary learning has been used in a real-time strategy (RTS) games (Ponsen and Spronck 2004). In Thurau et al. (2004), among others, human-like opponent behaviors are emerged through supervised-learning techniques in *Quake*. Even though complex opponent behaviors emerge, there is no further analysis of whether these behaviors contribute to the satisfaction of the player (i.e., interest of game). In other words, researchers hypothesize—by observing the vast number of multi-player online games played daily on the Web, for example—that by generating human-like opponents, they enable the player to gain more satisfaction from the game. This hypothesis might be true up to a point; however, since there is no explicit notion of interest defined, there is no evidence that a specific opponent behavior generates more or less interesting games. Such a

hypothesis is the core of Iida's work on board games. He proposed a general metric of entertainment for variants of chess games depending on average game length and possible moves (Iida et al. 2003).

## Identifying and Augmenting Entertainment

There have been several psychological studies to identify what is “fun” in a game and what engages people playing computer games. Theoretical approaches include Malone's (1981) principles of intrinsic qualitative factors for engaging game play, namely, challenge, curiosity, and fantasy as well as the well-known concepts of the theory of flow (Csikszentmihalyi 1990) incorporated in computer games as a model for evaluating player enjoyment, namely, *Game Flow* (Sweetser and Wyeth 2005). A comprehensive review of the literature on qualitative approaches for modeling player enjoyment demonstrates a tendency of overlapping with Malone's and Csikszentmihalyi's foundational concepts. Many of these approaches are based on Lazzaro's “fun” clustering, which uses four entertainment factors based on facial expressions and data obtained from game surveys on players (Lazzaro 2004): hard fun, easy fun, altered states, and socialization. Koster's (2005) theory of fun, which is primarily inspired by Lazzaro's four factors, defines “fun” as the act of mastering the game mentally. An alternative approach to fun capture is presented in Read et al. (2002), where fun is composed of three dimensions: durability, engagement, and expectations. Questionnaire tools and methodologies are proposed in order to empirically capture the level of fun for evaluating the usability of novel interfaces with children.

Work in the field of quantitative entertainment capture and augmentation is based on the hypothesis that the player-opponent interaction—rather than the audiovisual features, the context, or the genre of the game—is the property that contributes to the majority of the quality features of entertainment in a computer game (Yannakakis and Hallam 2004a). Based on this fundamental assumption, a metric for measuring the real-time entertainment value of predator/prey games was designed, and established as a generic interest metric for prey/predator games (Yannakakis and Hallam 2005a; 2005b). Further studies by Yannakakis and Hallam (2006) have shown that artificial neural networks (ANN) and fuzzy neural networks can extract a better estimator of player satisfaction than a human-designed one, given appropriate estimators of the challenge and curiosity of the game (Malone 1981) and data on human players' preferences. Similar work in adjusting a game's difficulty include endeavors through reinforcement learning (Andrade et al. 2005), genetic algorithms (Verma and McOwan 2005), probabilistic models (Hunicke and Chapman 2004), and dynamic scripting (Spronck et al. 2004). However, the aforementioned attempts

are based on the assumption that challenge is the only factor that contributes to enjoyable gaming experiences while results reported have not been cross-verified by human players.

A step further to entertainment capture is towards games of richer human-computer interaction and affect recognizers, which are able to identify correlations between physiological signals and the human notion of entertainment. Experiments by Yannakakis et al. (2006) have already shown a significant effect of the average heart rate of children's perceived entertainment in action games played in interactive physical playgrounds. Moreover, Rani et al. (2005) propose a methodology for detecting the anxiety level of the player and appropriately adjusting the level of challenge in the game of "Pong." Physiological state (heart-rate, galvanic skin response) prediction models have also been proposed for potential entertainment augmentation in computer games (McQuiggan et al. 2006).

We choose predator/prey games as the initial genre of our game research since, given our aims, they provide us with unique properties. In such games, we can deliberately abstract the environment and concentrate on the characters' behavior. The examined behavior is cooperative since cooperation is a prerequisite for effective hunting behaviors. Furthermore, we are able to easily control a learning process through online interaction. In other words, predator/prey games offer a well-suited arena for initial steps in studying cooperative behaviors generated by interactive online learning mechanisms. Even though other genres of games (e.g., FPS games) offer similar properties, researchers have not yet focused on the novel directions of human-verified entertainment capture and augmentation that are presented in this article.

## PREDATOR/PREY GAMES

Predator/prey games have been a very popular category of computer games and among its best representatives is the classical Pac-Man game released by Namco (Japan) in 1980. Even though Pac-Man's basic concept—the player's (*PacMan's*) goal is to eat all the pellets appearing in a maze-shaped stage while avoiding being killed by four opponent characters named "*Ghosts*"—and graphics are very simple, the game still keeps players interested after so many years, and its basic ideas are still found in many newly released games.

Kaiser et al. (1998) attempted to analyze emotional episodes, facial expressions, and feelings of humans playing a predator/prey computer game similar to Pac-Man (Kaiser and Wehrle 1996). Other examples in the Pac-Man domain literature include researchers attempting to teach a controller to drive *Pac-Man* in order to acquire as many pellets as possible and to avoid being eaten by *Ghosts*. Koza (1992) considers the problem of

controlling an agent in a dynamic nondeterministic environment and, therefore, sees Pac-Man as an interesting multi-agent environment for applying offline learning techniques based on genetic programming. Other approaches, such as incremental learning (Gallagher and Ryan 2003) and neuro-evolution (Lucas 2005), have also been applied for producing effective Pac-Man playing strategies. The same Pac-Man application domain has been used for analyzing size and generality issues in genetic programming (Rosca 1996).

On the other hand, there are many researchers who use predator/prey domains in order to obtain efficient emergent teamwork of either homogeneous or heterogeneous groups of predators. For example, Luke and Spector (1996), among others, have designed an environment similar to the Pac-Man game (the Serengeti world), in order to examine different breeding strategies and coordination mechanisms for the predators. Finally, there are examples of work in which both the predators' and the prey's strategies are co-evolved in continuous or grid-based environments (Haynes and Sen 1995; Miller and Cliff 1994).

Similar to Luke and Spector (1996), we view the domain from the predators' perspective and we attempt to emerge effective hunting teamwork offline based on evolutionary computation techniques applied to homogeneous neural controlled (Yao 1999) predators. However, playing a predator/prey computer game like Pac-Man against optimal hunters cannot be interesting because the player is consistently and effectively killed.

Researchers have generally shown that online learning in computer games is feasible through careful design and effective learning methodologies. On that basis, Yannakakis et al. ( 2004a; 2004c) introduced a neuro-evolution mechanism that acts in real time that optimizes each opponent individually (heterogeneous game environment) for generating appealing games rather than high opponent performance (Demasi and Cruz 2002; Graepel et al. 2004).

## **INTERESTING BEHAVIOR**

As noted, predator/prey games will be our test-bed genre for the investigation of enjoyable games. More specifically, in the games studied, the prey is controlled by the player and the predators are the computer-controlled opponents (nonplayer characters, or NPCs).

In the approach presented in this section, a quantitative metric of player satisfaction is designed based on general criteria of enjoyment. The first step towards generating enjoyable computer games is therefore to identify the criteria or features of games that collectively produce enjoyment (or else interest) in such games. Second, quantitative estimators for these criteria are defined and combined, in a suitable mathematical

formula, to give a single quantity correlated with player satisfaction (interest). Finally, this formulation player interest is tested against human players' judgment in real conditions using the Pac-Man test-bed.

Following the principles of Yannakakis and Hallam (2004a), we will ignore mechanics, audiovisual representation, control, and interface contributions to the enjoyment of the player and we will concentrate on the opponents' behaviors. A well-designed and popular game such as Pac-Man can fulfil all aspects of player satisfaction incorporated in the mentioned design game features. The player, however, may contribute to his/her entertainment through interaction with the opponents of the game and, therefore, it is implicitly included in the interest formulation presented here, see also Yannakakis and Maragoudakis (2005), for studies of the player's impact on his/her entertainment.

## Criteria

By observing the opponents' behavior in various predator/prey games, we attempted to identify the key features that generate entertainment for the player. These features were experimentally cross-validated against various opponents of different strategies and redefined when appropriate. According to Yannakakis and Hallam (2004a) and (2005b) the criteria that collectively define interest on any predator/prey game are briefly as follows.

1. Appropriate level of challenge (*when the game is neither too hard nor too easy*).
2. Behavior diversity (*when there is diversity in opponents' behavior over the games*).
3. Spatial diversity (*when opponents' behavior is aggressive rather than static*). That is, predators that move constantly all over the game world and cover it uniformly. This behavior gives the player the impression of an intelligent strategic opponents' plan, which increases the game interest.

## Metrics

In order to estimate and quantify each of the three aforementioned interest criteria, we let the examined group of opponents play the game  $N$  times (each game for a sufficiently large evaluation period of  $t_{\max}$  simulation steps) and we record the simulation steps  $t_k$  taken to kill the player, as well as the total number of the opponents' visits  $v_{ik}$  at each cell  $i$  of the grid game field for each game  $k$ . In the case where the game's motion is continuous, a discretization of the field's plane up to the character's size can serve this purpose.

Given these, the quantifications of the three interest criteria proposed above can be presented as follows.

1. *Appropriate Level of Challenge:* According to the first criterion, an estimate of how interesting the behavior is, is given by  $T$  in (1).

$$T = [1 - (E\{t_k\} / \max\{t_k\})]^{p_1}, \quad (1)$$

where  $E\{t_k\}$  is the average number of simulation steps taken to kill the prey-player over the  $N$  games;  $\max\{t_k\}$  is the maximum  $t_k$  over the  $N$  games; and  $p_1$  is a weighting parameter.

$p_1$  is adjusted so as to control the impact of the bracketed term in the formula for  $T$ . By selecting values of  $p_1 < 1$ , we reward quite challenging opponents more than near-optimal killers, since we compress the  $T$  value toward 1. More details on the adjustment of the  $p_1$  value for the Pac-Man game will follow.

The  $T$  estimate of interest demonstrates that the greater the difference between the average and the maximum number of steps taken to kill the player, the higher the interest of the game. Given (1), both easy killing (“*too easy*”) and near-optimal (“*too hard*”) behaviors receive low interest estimate values (i.e.,  $E\{t_k\} \simeq \max\{t_k\}$ ). This metric is also called “challenge.”

2. *Behavior Diversity:* The interest estimate for the second criterion is given by  $S$  in (2).

$$S = (\sigma/\sigma_{max})^{p_2}, \quad (2)$$

where

$$\sigma_{max} = \frac{1}{2} \sqrt{\frac{N}{(N-1)} (t_{max} - t_{min})} \quad (3)$$

and  $\sigma$  is the standard deviation of  $t_k$  over the  $N$  games;  $\sigma_{max}$  is an estimate of the maximum value of  $\sigma$ ;  $t_{min}$  is the minimum number of simulation steps required for predators to kill the prey obtained by playing against some “well”-behaved fixed strategy near-optimal predators ( $t_{min} \leq t_k$ ); and  $p_2$  is a weighting parameter.

The  $S$  increases proportionally with the standard deviation of the steps taken to kill the player over  $N$  games. Therefore, using  $S$  as defined here, we promote predators that produce high diversity in the time taken to kill the prey.

3. *Spatial Diversity:* A good measure for quantifying the third interest criterion is through entropy of the predators’ cell visits in a game, which

quantifies the completeness and uniformity with which the opponents cover the stage. Hence, for each game, the cell visit entropy is calculated and normalized into [0, 1] via (4).

$$H_n = \left[ -\frac{1}{\log V_n} \sum_i \frac{v_{in}}{V_n} \log \left( \frac{v_{in}}{V_n} \right) \right]^{p_3}, \quad (4)$$

where  $V_n$  is the total number of visits of all visited cells (i.e.,  $V_n = \sum_i v_{in}$ ) and  $p_3$  is a weighting parameter.  $p_3$  is adjusted in order to promote very high  $H_n$  values and furthermore to emphasize the distinction between high and low normalized entropy values. Appropriate  $p_3$  parameter values, which serve this purpose, are those greater than one ( $p_3 = 4$  in this article), since they stretch the value of  $H_n$  away from 1.

Given the normalized entropy values  $H_n$  for all  $N$  games, the interest estimate for the third criterion can be represented by their average value  $E\{H_n\}$  over the  $N$  games. This implies that the higher the average entropy value, the more interesting the game is.

The three individual criterion metrics defined are combined linearly to produce a single metric of interest (Equation 5) whose properties match the qualitative considerations developed.

$$I = \frac{\gamma T + \delta S + \epsilon E\{H_n\}}{\gamma + \delta + \epsilon}, \quad (5)$$

where  $I$  is the interest value of the predator/prey game;  $\gamma$ ,  $\delta$  and  $\epsilon$  are criterion weight parameters.

The interest metric introduced in (5) can be applied effectively to any predator/prey computer game, because it is based on generic features of this category of games (see Yannakakis and Hallam [2004b; 2005] for successful applications to dissimilar predator/prey games). These features include the time required to kill the prey and the predators' entropy throughout the game field. We therefore believe that (5)—or a similar measure of the same concepts—constitutes a generic interest approximation of predator/prey computer games. Moreover, given the two first interest criteria previously defined, the approach's generality is expandable to all computer games. Indeed, no player likes any computer game that is too difficult or too easy to play and, furthermore, any player would enjoy diversity throughout the play of any game. The third interest criterion is applicable to games where spatial diversity is important which, apart from predator/prey games, may also include action, strategy, and team sports games

according to the computer game genre classification of Laird and van Lent (2000).

The approach to entertainment modeling represented by equation (5) is both innovative and efficient. However, it should be clear that there are many possible formulae, such as equation (5), which would be consistent with the qualitative criteria proposed for predator/prey games. Other successful quantitative metrics for the appropriate level of challenge, the opponents' diversity and the opponents' spatial diversity may be designed and more qualitative criteria may be inserted in the interest formula. Alternative mathematical functions for combining and weighting the various criteria could be employed.

For example, other metrics for measuring the appropriate level of challenge could be used: One could come up with a  $T$  metric assuming that the appropriate level of challenge follows a Gaussian distribution over  $E\{t_k\}$  and that the interest value of a given game varies, depending on how long it is—*very short* ( $E\{t_k\} \approx t_{min}$ ) games tend to be frustrating and *long games* ( $E\{t_k\} \approx \max\{t_k\}$ ) tend to be boring. (However, very short games are not frequent in the experiments presented here and, therefore, by varying the weight parameter  $p_1$  in the proposed  $T$  metric [see (1)], we are able to obtain an adequate level of variation in measured challenge.)

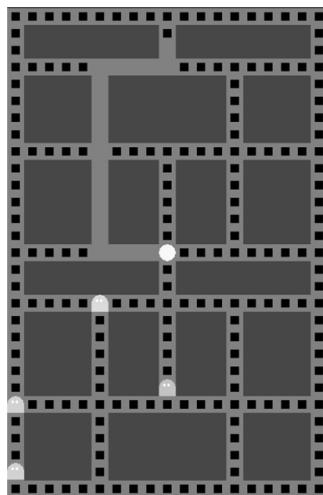
The question remains, however, whether the number produced by such a formula really captures anything useful concerning a notion so potentially complex as human enjoyment. That question is addressed next.

## THE PAC-MAN TEST-BED

The computer game test-bed studied is a modified version of the original Pac-Man computer game released by Namco. The player's (*Pac-Man's*) goal is to eat all the pellets appearing in a maze-shaped stage, while avoiding being killed by the four *Ghosts*. The game is over when either all pellets in the stage are eaten by *Pac-Man*, *Ghosts* manage to kill *Pac-Man*, or a pre-determined number of simulation steps is reached without any of these occurring. In that case, the game restarts from the same initial positions for all five characters.

Compared to commercial versions of the game, a number of features (e.g., power pills) are omitted for simplicity; these features do not qualitatively alter the nature of “interesting” in games of low interest. Cross-validation of this statement appears through the judgment and the beliefs of human players of both the original and this version of the game.

As stressed previously, the Pac-Man game is investigated from the viewpoint of *Ghosts* and more specifically how *Ghosts'* emergent adaptive behaviors can contribute to the interest of the game. Pac-Man—as a computer game domain for emerging adaptive behaviors—is a two-dimensional,



**FIGURE 1** Snapshot of the Pac-Man game.

multi-agent, grid-motion, predator/prey game. The game field (i.e., stage) consists of corridors and walls. Both the stage's dimensions and its maze structure are predefined. For the experiments presented in this article, we use a  $19 \times 29$  grid maze-stage where corridors are one grid-cell wide (see Figure 1).

The characters visualized in the Pac-Man game (as illustrated in Figure 1) are a white circle that represents *Pac-Man* and four ghost-like characters representing the *Ghosts*. Being *Ghosts*, one of their properties is permeability, i.e., two or more *Ghosts* can simultaneously occupy the same cell of the game grid. Additionally, there are black squares that represent the pellets and dark gray blocks of walls.

*Pac-Man* moves at double the *Ghosts'* speed and since there are no dead ends, it is impossible for a single *Ghost* to complete the task of killing it. Since *Pac-Man* moves faster than a *Ghost*, the only effective way to kill a well-performing *Pac-Man* is for a group of *Ghosts* to hunt cooperatively.

### Pac-Man

Both the difficulty and, to a lesser degree, the interest of the game are directly affected by the intelligence of the *Pac-Man* player. In Yannakakis and Hallam (2004a; 2005), three fixed *Ghost*-avoidance and pellet-eating strategies for the *Pac-Man* player, differing in complexity and effectiveness, are presented. Each strategy is based on decision making applying a cost or probability approximation to the player's four neighboring cells (i.e., up, down, left, and right). We present them briefly in this article.

- Cost-Based (CB) *Pac-Man*: Moves towards a cost minimization path that produces effective *Ghost*-avoidance and (to a lesser degree) pellet-eating behaviors, but only in the local neighbor cell area.
- Rule-Based (RB) *Pac-Man*: This is a CB *Pac-Man*, plus an additional rule for more effective and global pellet-eating behavior.
- Advanced (ADV) *Pac-Man*: The ADV moving strategy generates a more global *Ghost*-avoidance behavior built upon the RB *Pac-Man*'s good pellet-eating strategy.

### **Ghosts**

The arcade version of Pac-Man uses a handful of simple rules and scripted sequences of actions that control each *Ghost* individually (Wikipedia), combined with some random decision making to make the *Ghosts*' behavior less predictable. Even though this design yields quite complex *Ghost* behaviors, the player's interest decreases at the point where *Ghosts* are too fast to beat (Rabin 2002). In our Pac-Man version, we require *Ghosts* to keep learning and constantly adapting to the player's strategy, instead of being opponents with fixed strategies and furthermore maintain a constant real-time speed.

Neural networks are a suitable host for emergent adaptive behaviors in complex multi-agent environments (Ackley and Littman 1992) and have been successfully applied for adaptive learning in real time in computer games (Stanley et al. 2005). A multi-layered fully connected feedforward neural controller, where the sigmoid function is employed at each neuron, manages the *Ghosts*' motion. Using their sensors, *Ghosts* inspect the environment from their own point of view and decide their next action. Each *Ghost*'s perceived input consists of the relative coordinates of *Pac-Man* and the closest *Ghost*. We deliberately exclude from consideration any global sensing, e.g., information about the dispersion of the *Ghosts* as a whole, because we are interested specifically in the minimal sensing scenario. The neural network's output is a four-dimensional vector with respective values from 0 to 1 that represents the *Ghost*'s four movement options (up, down, left, and right, respectively). Each *Ghost* moves towards the available—unobstructed by walls—direction represented by the highest output value. Available movements include the *Ghost*'s previous cell position.

### **Fixed Strategy Ghosts**

Apart from the neural controlled *Ghosts*, three additional non-evolving strategies have been tested for controlling the *Ghost*'s motion. These strategies are used as baseline behaviors for comparison with any neural-controller emerged behavior.

- Random (R): *Ghosts* that randomly decide their next available movement. Available movements have equal probabilities to be picked.
- Followers (F): *Ghosts* designed to follow *Pac-Man* constantly. Their strategy is based on moving so as to reduce the greatest of their relative coordinates from *Pac-Man*.
- Near-Optimal (O): A *Ghost* strategy designed to produce attractive forces between *Ghosts* and *Pac-Man*, as well as repulsive forces among the *Ghosts*.<sup>1</sup>

## ADJUSTING INTEREST PARAMETER VALUES FOR PAC-MAN

In this section, we present the procedures followed to obtain the appropriate parameter values of the interest estimate (5) for the Pac-Man game. In this article  $t_{\min}$  is 35 simulation steps, which is obtained as the minimum simulation time that *Pac-Man* survives when playing against the best-performing near-optimal *Ghosts*. In addition,  $t_{\max}$  is 320 simulation steps, which corresponds to the minimum simulation period required by the RB *Pac-Man* (best pellet-eater) to clear the stage of pellets.

In order to obtain values for the interest formula weighting parameters  $p_1$ ,  $p_2$ , and  $p_3$ , we select empirical values based on each interest criterion. For the first interest metric presented in (1),  $p_1$  is adjusted so as to give  $T$  a greater impact or else a boost when even a slight difference between the maximum and the average lifetime of the player (i.e., challenge) is noted ( $p_1 < 1$ ). This way we reward quite challengeable opponents more than near-optimal killers. For the third interest metric presented in (4),  $p_3$  is adjusted in order to press for very high  $H_n$  values and furthermore to provide a clearer distinction between high and low normalized entropy values ( $p_3 > 1$ ). Finally,  $p_2$  is set so as  $\sigma$  has a linear effect on  $S$ . By taking this into consideration,  $p_1 = 0.5$ ,  $p_2 = 1$ , and  $p_3 = 4$ , for the experiments presented in this article.

Moreover, values for the interest criteria weighting parameters  $\gamma$ ,  $\delta$ , and  $\epsilon$  are also selected empirically based on the specific game. In Pac-Man, aggressive opponent behavior is of the greatest interest. The game loses any reliability when *Ghosts* stick in a corner instead of wandering around the stage. Thus, diversity in gameplay ( $S$ ) and appropriate level of challenge ( $T$ ) should come next in the importance list of interest criteria. Given the previously mentioned statements and by adjusting these three parameters so that the interest value escalates as the opponent behavior changes from Random to near-optimal, and then to follower, we come up with  $\gamma = 1$ ,  $\delta = 2$  and  $\epsilon = 3$ .

Since the interest value changes monotonically with respect to each of the three criterion values  $T$ ,  $S$ ,  $E\{H_n\}$ , sensitivity analysis is conducted on

the interest metric parameters, aiming to portray the relation between these parameters as well as their weighting degree in the interest formula. We therefore proceed by seeking opponent behaviors that generate ten different  $T$ ,  $S$ , and  $E\{H_n\}$  values, equally spread in the  $[0,1]$  interval. Given these thirty values as input,  $p_1$ ,  $p_2$ ,  $p_3$ ,  $\gamma$ ,  $\delta$ , and  $\epsilon$  parameters are systematically changed one at a time so that their percentage difference lies in the interval  $[-50\%, 50\%]$ . Each time a parameter change occurs, the absolute percentage difference of the game's interest is computed. The function between the absolute percentage differences of the interest value and the percentage differences of the interest weighting parameters is illustrated in Figure 2.

As seen in Figure 2, changes on the  $p_1$ ,  $p_2$ , and  $p_3$  parameters seem to influence the  $I$  value more than  $\gamma$ ,  $\delta$ , and  $\epsilon$ . The observed difference in interest sensitivity is reasonable since the first three parameters represent powers, while the latter three correspond to product weights. More specifically,  $p_2$  and  $p_3$  reveal significant differences (i.e., greater than 5%) in  $I$  when decreased by 15% (i.e.,  $p_2 = 0.85$ ) and 9% (i.e.,  $p_3 = 3.64$ ) or increased by 20% (i.e.,  $p_2 = 1.2$ ) and 10% (i.e.,  $p_3 = 4.4$ ), respectively. For  $p_1$  significant change in  $I$  is observed only when decreased by up to 35% (i.e.,  $p_1 = 0.325$ ). Accordingly, both  $\epsilon$  and  $\delta$  parameters reveal significant differences in  $I$  only when decreased by 40% and 45% respectively. Finally, for  $\gamma$  no significant change in  $I$  is observed even when changed by up to 50%.

Regardless of the sensitivity of the  $I$  value, as far as mainly the  $p_2$  and  $p_3$  parameters are concerned, we believe that the selected values project a robust  $I$  value considering that they constitute power parameters in the interest formula.

## MEASURING PERFORMANCE

When a predator/prey game is investigated from the predator's viewpoint, optimality can be measured in the predators' ability to kill the prey. Thus, prey-killing ability of the *Ghosts* is the primary factor that determines how well-performed a behavior is in the *Pac-Man* game. Furthermore, preventing *Pac-Man* from eating pellets, which also implies fast-killing capabilities, constitutes an additional factor of the desired optimal behavior. Given these, a measure designed to give an approximation of a group of *Ghosts*' performance over a specific number  $N$  of games played, is

$$P = \frac{\alpha(k/N) + \beta \min\{1 + (e_{\min} - E\{e\})/e_{\max} - e_{\min}), 1\}}{\alpha + \beta}, \quad (6)$$

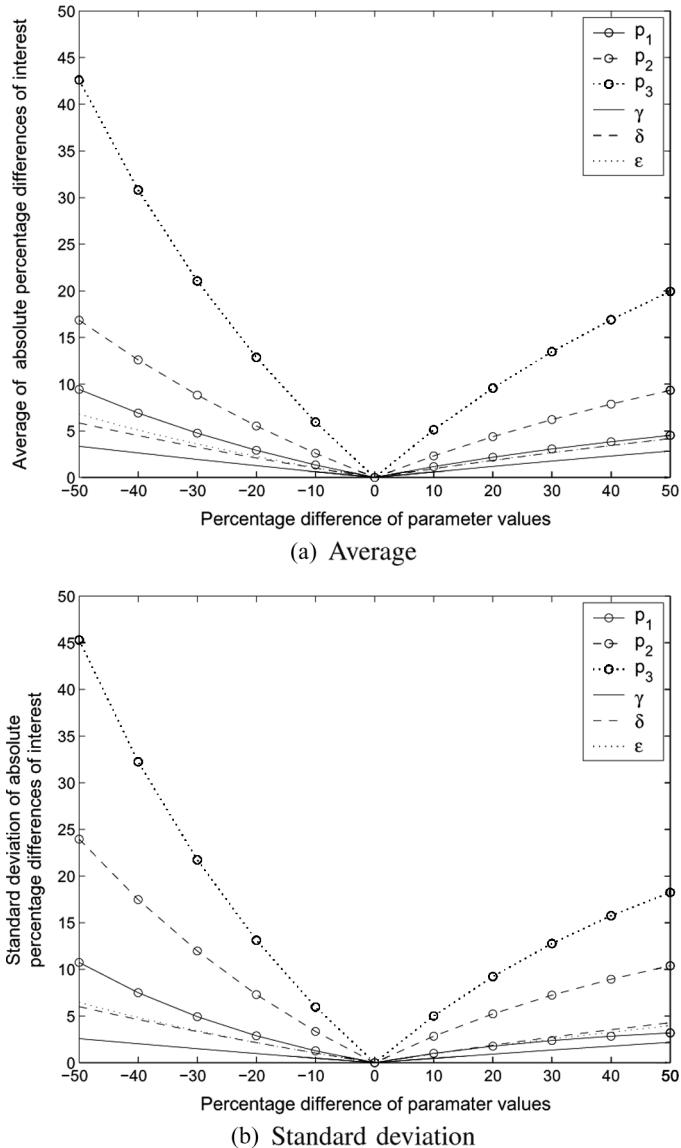


FIGURE 2 Absolute percentage differences of  $I$  over ten runs for each weighting parameter.

where  $P$  is the performance of a *Ghost* group behavior taking values from 0 to 1;  $k$  is the number of *Pac-Man* kills within  $N$  games;  $E\{e\}$  is the average number of pellets eaten by *Pac-Man* over the  $N$  games;  $e_{\min}$ ,  $e_{\max}$  are the lower and upper bound of the eaten pellets  $e$ , respectively (in this article  $e_{\min} = 80$ ,  $e_{\max} = 227$ );  $\alpha$ ,  $\beta$  are weight parameters (in this article  $\alpha = \beta = 1$ ).

## OFFLINE LEARNING

We use an offline evolutionary learning approach in order to produce some “good” (i.e., in terms of performance) initial behaviors. An additional aim of the proposed algorithm is to emerge dissimilar behaviors of high fitness—varying from blocking to aggressive—offering diverse seeds for the online learning mechanism in its attempt to generate emergent *Ghost* behaviors that make the game interesting. The offline learning mechanism used is presented in Yannakakis and Hallam (2004a) and Yannakakis et al. (2004c).

## ONLINE LEARNING (OLL)

As previously noted, games which can learn and adapt to new playing strategies offer a richer interaction to entertain the player. For that purpose, we use an evolutionary machine learning mechanism for the Pac-Man game, which is based on the idea of *Ghosts* that learn while they are playing against *Pac-Man*. Or else, *Ghosts* that are reactive to any player’s behavior learn from its strategy instead of being opponents with fixed strategies that exist in all versions of this game today. Furthermore, this approach’s additional objectives are to keep the game’s interest at high levels as long as it is being played and to achieve good real-time performance (i.e., low computational effort during gameplay). This approach is first introduced in Yannakakis et al. (2004a) for a prey-predator game called “Dead-End” and in (Yannakakis and Hallam (2004a) for the Pac-Man game.

Beginning from any initial group of OLT *Ghosts*, the OLL mechanism transforms them into a group of heterogeneous opponents that are conceptually more interesting to play against. An OLT *Ghost* is cloned four times and its clones are placed in the game field to play against a selected fixed *Pac-Man* type in a selected stage. Then, at each generation:

Step 1. Each *Ghost* is evaluated every  $t$  simulation steps via (7), while the game is played— $t = 50$  simulation steps in this article.

$$f' = \sum_{i=1}^{t/2} \{d_{P,i} - d_{P,(2i-1)}\}, \quad (7)$$

where  $d_{P,i}$  is the distance between the *Ghost* and *Pac-Man* at the  $i$  simulation step. This fitness function promotes *Ghosts* that move towards *Pac-Man* within an evaluation period of  $t$  simulation steps.

- Step 2. A pure elitism selection method is used where only the fittest solution is able to breed. The fittest parent clones an offspring with a probability  $p_c$  that is inversely proportional to the normalized cell visit entropy (i.e.,  $p_c = 1 - H_n$ ) given by (4). In other words, the higher the cell visit entropy of the *Ghosts*, the lower the probability of breeding new solutions. If there is no cloning, then go back to Step 1, else continue to Step 3.
- Step 3. Mutation occurs in each gene (connection weight) of the offspring's genome with a small probability  $p_m$  (e.g., 0.02). A Gaussian random distribution is used to define the mutated value of the connection weight. The mutated value is obtained from (8).

$$w_m = \mathcal{N}(w, 1 - H_n), \quad (8)$$

where  $w_m$  is the mutated connection weight value and  $w$  is the connection weight value to be mutated. The Gaussian mutation, presented in (8), suggests that the higher the normalized entropy of a group of *Ghosts*, the smaller the variance of the Gaussian distribution and therefore, the less disruptive the mutation process as well as the finer the precision of the GA.

- Step 4. The mutated offspring is evaluated briefly via (7) in offline mode, that is, by replacing the least-fit member of the population and playing an offline (i.e., no visualization of the actions) short game of  $t$  simulation steps. If there is a human playing Pac-Man, then the *Pac-Man*'s motion trail of the last  $t$  simulation steps is recorded and opponents are evaluated against it in offline mode. The fitness values of the mutated offspring and the least-fit *Ghost* are compared and the better one is kept for the next generation. This pre-evaluation procedure for the mutated offspring attempts to minimize the probability of group behavior disruption by low-performance mutants. The fact that each mutant's behavior is not tested in a single-agent environment but within a group of heterogeneous *Ghosts*, helps more towards this direction. If the least-fit *Ghost* is replaced, then the mutated offspring takes its position in the game field as well.

The algorithm is terminated when a predetermined number of games has been played or a game of high interest (e.g.,  $I \geq 0.7$ ) is found.

We mainly use short simulation periods ( $t = 50$ ) in order to evaluate *Ghosts* in OLL, aiming to the acceleration of the online evolutionary process. The same period is used for the evaluation of mutated offspring; this is based on two primary objectives: 1) to apply a fair comparison between the mutated offspring and the least-fit *Ghost* (i.e., same evaluation period) and

2) to avoid undesired high computational effort in online mode (i.e., while playing).

## EXPERIMENTS AGAINST FIXED PLAYING STRATEGIES

Results obtained from experiments applied on the *Pac-Man* game against fixed playing strategies are presented in this section. These include offline training and online learning emergent behavior analysis as well as experiments for testing robustness and adaptability of the online learning approach proposed.

### Offline Training

The procedure presented in this subsection is focused on generating well-behaved *Ghosts* in terms of the performance measure described previously. We train *Ghosts* against all three types of *Pac-Man* player through the neuro-evolution offline learning mechanism presented in previously.

In order to minimize the non-deterministic effect of the *Pac-Man*'s strategy on the *Ghost's* performance and interest values as well as to draw a clear picture of these averages' distribution, we apply the following bootstrapping procedure. Using a uniform random distribution we pick 10 different 50-tuples out of the 100 previously mentioned games. These 10 samples of data (i.e.,  $e, k, t_k, v_{ik}$ ) from 50 games (i.e.,  $N = 50$ ) are used to determine the *Ghost's* average performance and interest values and their respective confidence intervals. The outcome of this experiment is presented in Table 1.

**TABLE 1** Performance ( $P$ ) and Interest ( $I$ ) Values (Average Values of 10 Samples of 50 Games Each) of Fixed Strategy (R, F, O) and OLT *Ghosts* (B, A, H) Playing against All Three *Pac-Man* Types (CB, RB, ADV). Average  $P$  and  $I$  Values ( $E\{\cdot\}$ ) of All Six Strategies Appear in the Bottom Row. Experiment Parameters: Population Size is 80,  $g = 1000$ ,  $t = 320$  Simulation Steps.  $N_t$  = Games,  $p_m = 0.02$ , 5-Hidden Neurons Controller

| Trained offline by playing against |       |       |       |       |       |       |
|------------------------------------|-------|-------|-------|-------|-------|-------|
|                                    | CB    |       | RB    |       | ADV   |       |
|                                    | $P$   | $I$   | $P$   | $I$   | $P$   | $I$   |
| R                                  | 0.423 | 0.547 | 0.363 | 0.586 | 0.356 | 0.523 |
| F                                  | 0.754 | 0.771 | 0.701 | 0.772 | 0.621 | 0.771 |
| O                                  | 0.891 | 0.729 | 0.897 | 0.749 | 0.964 | 0.686 |
| B                                  | 0.734 | 0.576 | 0.689 | 0.412 | 0.869 | 0.442 |
| A                                  | 0.661 | 0.654 | 0.606 | 0.652 | 0.662 | 0.555 |
| H                                  | 0.348 | 0.190 | 0.310 | 0.250 | 0.467 | 0.423 |
| $E\{\cdot\}$                       | 0.635 | 0.578 | 0.592 | 0.570 | 0.656 | 0.566 |

Offline trained emergent solutions are the OLL mechanisms' initial points in the search for more interesting games. OLT obtained behaviors are classified into the following categories.

- Blocking (B): These are the OLT *Ghosts* that achieve the best performance against each *Pac-Man* type. Their behavior is characterized as ‘Blocking’ because they tend to wait for *Pac-Man* to enter a specific area that is easy for them to block and then kill. Their average normalized cell visit entropy value  $E\{H_n\}$  lies between 0.55 and 0.65.
- Aggressive (A): These are OLT *Ghosts* that achieve lower performance in comparison to the blockers. Their behavior is characterized as “aggressive” because they tend to follow *Pac-Man* all over the stage in order to kill it. This motion feature generates the highest  $I$  value ( $E\{H_n\} \geq 0.65$ ) among the interest values generated by the three emergent behaviors.
- Hybrid (H): These are suboptimal OLT *Ghosts* that achieve the lowest performance ( $P \leq 0.55$ ) and low interest value in comparison to the aforementioned B and A *Ghosts* ( $E\{H_n\} < 0.55$ ). Their behavior is characterized as “hybrid” because they tend to behave as a blocking-aggressive hybrid, which proves to be ineffective at killing *Pac-Man*.

As far as the interest value generated by the mentioned behaviors is concerned, confidence intervals ( $\pm 0.0647$  maximum,  $\pm 0.0313$  on average) obtained by the bootstrapping procedure previously described indicate that B, A, and H are significantly different.

According to Table 1, near-optimal and blocking behavior *Ghosts* achieve high-performance values against all three *Pac-Man* types, whereas their interest value is not as high as their performance value. This reveals the compromise between optimality and interest it has to be made because, in a predator/prey computer game, optimal killing behaviors are almost never interesting behaviors. On the other hand, followers are likely to produce the most interesting behaviors (among the behaviors examined in Table 1) for the game.

Viewing results presented in Table 1 from the *Pac-Man* type perspective (i.e., the average values in the bottom row of the table), it looks as if the RB and ADV are, respectively, the hardest and easiest *Pac-Man* players to kill. Concerning the three *Pac-Man* types' generated interest, it seems that there is no significant difference amongst them.

## Online Learning Experiments

As previously mentioned, the offline learning procedure is a mechanism that produces near-optimal solutions to the problem of killing

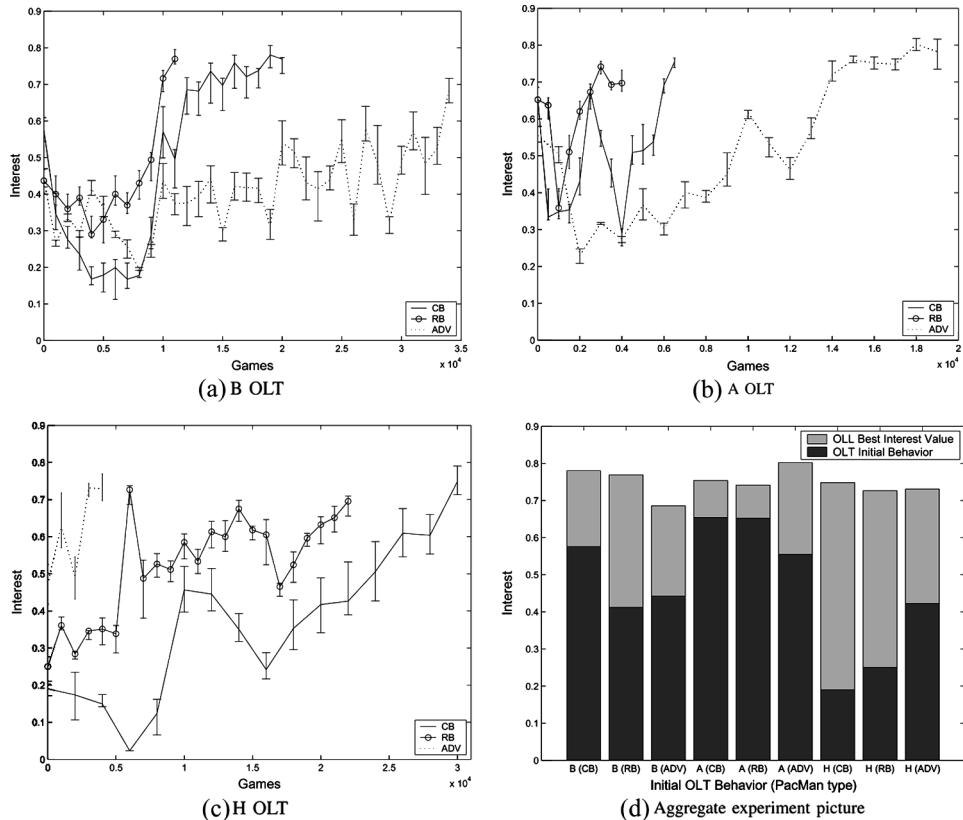
*Pac-Man* and minimizing the pellets eaten in a game. These solutions are the OLL mechanisms' initial points in the search for more interesting games. The OLL experiment is described as follows.

- Pick the nine emerged *Ghosts*' behaviors produced from the offline learning experiments presented previously (i.e., B, A and H behaviors emerged by playing against each *Pac-Man* type).
- Starting from each OLT behavior, apply the OLL mechanism by playing against the same type of *Pac-Man* as was used offline.
- Calculate the interest (bootstrapping procedure with  $N = 50$  of the game every 500 games during each OLL attempt.

The evolution of interest over the OLL games of each one of the three OLT behaviors is presented in a subfigure of Figure 3. For each of the three subfigures, three lines are illustrated, representing the interest values and their respective confidence intervals of the OLL attempt playing against the three *Pac-Man* types. Figure 3(d) illustrates the overall picture of the OLL experiments by comparing the initial interest of the game against the best average interest value achieved from OLL. Clearly, the OLL approach constitutes a robust mechanism that, starting from near-optimal or suboptimal *Ghosts*, manages to emerge interesting games (i.e., interesting *Ghosts*) in the vast majority of cases (i.e., in 8 out of 9 cases  $I > 0.7$ ). In 5 out of 9 OLL attempts, the best interest value is significantly greater or statistically equal to the respective follower's value (i.e., 0.771 against CB, 0.772 against RB and 0.771 against ADV).

As seen from Figure 3, OLL enhances the interest of the game independently of the initial OLT behavior or the *Pac-Man* player *Ghosts* play against. In all experiments presented here, the learning mechanism is capable of producing games of higher than the initial interest as well as keeping that high interest for a long period. There is obviously a slight probability of disruptive mutations (the higher the game's interest, through the cell visit entropy value, the less the probability of mutation) that can cause undesired drops in the game's interest. However, OLL is robust enough to recover from such disruptive phenomena (Figure 3).

Given an interesting initial behavior (e.g., aggressive behavior,  $I > 0.6$ ), it takes some few thousands of games for the learning mechanism to produce games of high interest. On the other hand, it takes some several thousand games to transform an uninteresting near-optimal blocking behavior (see Figure 3(a) and Figure 3(c)) into an interesting one. That is because the OLL process requires an initial long period to disrupt the features of an uninteresting blocking behavior, in order to be able to boost the interest of the game.



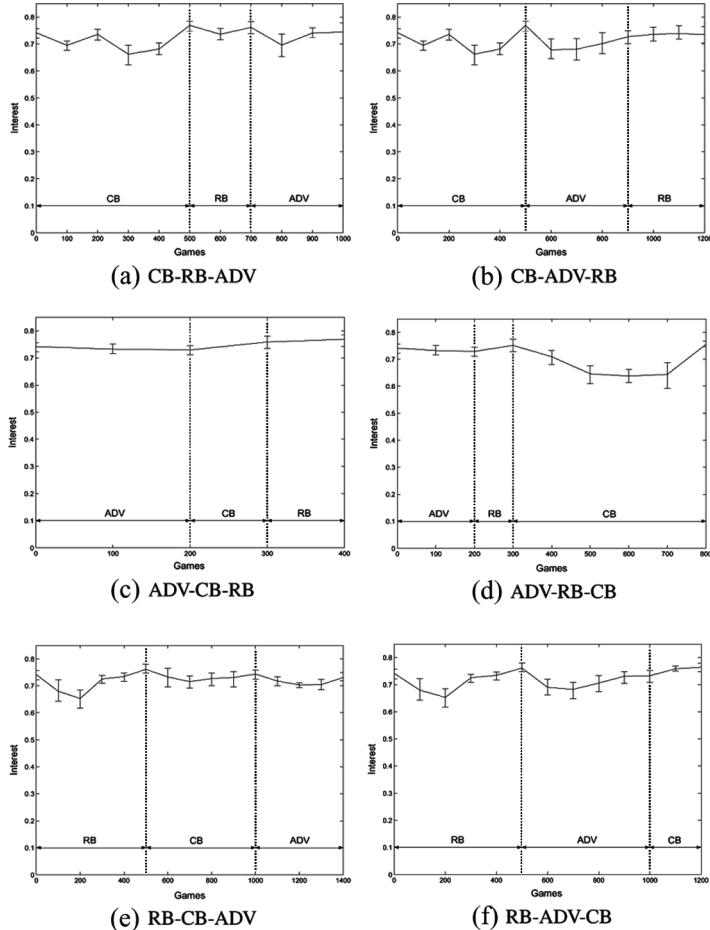
**FIGURE 3** Game interest over the number of OLL games. For reasons of computational effort, the OLL procedure continues for a number of games large enough to illustrate its behavior, after a game of high interest ( $I \geq 0.7$ ) is found. Initial *Ghost* behaviors appear in (a), (b), and (c) subfigure caption, whereas (d) illustrates the overall picture of the experiment. Experiment parameters:  $t = 50$  simulation steps,  $p_m = 0.02$ , 5-hidden neurons controller.

It is obvious that a number in the scale of  $10^3$  constitutes an unrealistic number of games for a human player to play. On that basis, it is very unlikely for a human to play so many games in order to notice the game's interest increasing. The reason for the OLL process being that slow is a matter of keeping the right balance between the process' speed and its "smoothness" (by smoothness we define the interest's magnitude of change over the games). A solution to this problem is to consider the initial long period of disruption as an offline learning procedure and start playing as soon as the game's interest is increased. Moreover, other online learning approaches like co-evolution (Demasi and Cruz 2002), dynamic scripting (Spronck et al. 2004), and reinforcement learning (Graepel et al. 2004; Andrade et al. 2005) could, in part, provide a solution for the long convergence times observed.

How effective will this mechanism be in a potential change from a fixed strategy to a dynamic human *Pac-Man* player? The next subsection provides evidence in order to support the answer.

## Adaptability

In order to test the OLL approach's ability to adapt to a changing environment (i.e., change of *Pac-Man* strategy), the following experiment is proposed. Beginning from an initial behavior of high interest value  $I_{init}$ , we apply the OLL mechanism against a specific *Pac-Man* type. During the online process, we keep changing the type of player as soon as interesting



**FIGURE 4** Online learning *Ghosts* playing against changing types of *Pac-Man*. Sub-figure captions indicate the playing *Pac-Man* sequence.

games (i.e.,  $I \geq I_{init}$ ) are produced. The process stops when all three types of players have played the game.

Since we have three types of players, the total number of such experiments is six (all different player type sequences). These experiments illustrate the overall picture of the approach's behavior against any sequence of *Pac-Man* types. As seen in Figure 4, OLL is able to quickly recover a sudden change in the player's strategy and boost the game's interest at high levels after sufficient games have been played (i.e., 100 to 500 games). The mechanism demonstrates a similar adaptive behavior for all six sequences of *Pac-Man* players, which illustrates its independence of the sequence of the changing *Pac-Man* type.

Results obtained from this experiment provide evidence for the approach's ability to adapt to new types of players as well as its efficiency in producing interesting games against humans with dynamic playing strategies.

## EXPERIMENTS AGAINST HUMAN PLAYERS

Experiments against fixed playing strategies portrayed the OLL mechanism's ability to generate interesting *Pac-Man* games. Apart from being fairly robust, the proposed mechanism demonstrated high and fast adaptability to changing types of player (i.e., playing strategies). The next obvious step to take is to let humans judge whether generated games are realistically interesting, or not, and whether OLL indeed enhances the level of entertainment during play. For this, we conducted a survey, with human subjects as *Pac-Man* players, that primarily aimed to obtain answers for the following questions.

1. Does the interest value computed for a game correlate with human judgment of interest?
2. Does the online learning mechanism cause perceived interest to change? Do perceived changes match computed ones?

The experiment is described next. Then the statistical method used and the analysis of obtained results are presented, respectively.

## EXPERIMENT DESCRIPTION

Answers to the target questions presented previously are based on statistical analysis of data acquired from a questionnaire applied for the *Pac-Man* game. The main prerequisite for a subject to participate in this experiment is to have played the original version (Namco) of the *Pac-Man* game at least once. Subject age covers a range between 17 and 51

years, where both sexes are almost equally represented (43.3% females, 56.7% males). In addition, all subjects speak English as a foreign language since their nationality is either Danish (90%) or Greek (10%). The questionnaire is divided into three parts (A, B, and C) and the steps that the subjects go through at each part are presented as follows.

## Personal Data

Subjects are asked to define their interest in the Pac-Man game before they play it. Answers categorize participants into three types of Pac-Man player' represented as "Like," "Neutral," and "Don't like."

Subjects familiarize themselves with the game by playing 50 games against specific OLT opponents (i.e., opponent 4 presented in Table 2). Online learning is used during this testing period, which is not noticeable to the player. At the end of the testing period, each subject's opponents trained online are saved.

## 1st Objective

We pick opponents differing in interest measured against the ADV player (as the most advanced computer-guided *Pac-Man* player). We select five opponents whose interest values uniformly cover the [0, 1] space. The selected opponents' number, which is used as an id-code, and their respective interest values are presented in Table 2.

By experimental design, each subject plays against three of the selected opponents in all permutations of pairs. In addition, we require equal participation of all three player types. For this experiment, we use 30 subjects divided into three equal subsets for each of the three player types (Like, Neutral, Don't Like), since  $C_3^5 = 10$  subjects are required for each player type. Moreover, observed effects show that 30 subjects constitute a statistically significant sample.

**TABLE 2** The Selected Opponents and Their Respective Interest— $I$  and 95% Confidence Interval ( $I_u$ ,  $I_l$ ) Values

| Opponent | Interest |        |        |
|----------|----------|--------|--------|
|          | $I_u$    | $I$    | $I_l$  |
| 1        | 0.2043   | 0.1793 | 0.1494 |
| 2        | 0.3673   | 0.3158 | 0.2670 |
| 3        | 0.5501   | 0.4943 | 0.4420 |
| 4        | 0.6706   | 0.6484 | 0.6267 |
| 5        | 0.8180   | 0.8023 | 0.7858 |

- As previously mentioned, each subject plays sets of games (five games in each set) against three of the selected opponents in all permutations of pairs and each time a pair of sets is completed, the player is asked whether the first set was more interesting than the second set of games. We use the 2-alternative forced choice (2-AFC) approach since it offers several advantages for a subjective interest capture. The 2-AFC comparative fun analysis (Read et al. 2002) minimizes the assumptions about people's different notions of entertainment and provides data for a fair comparison among answers of different people (Yannakakis et al. 2006)

The total number of sets of games that is played by each subject is 12 (all permutations of three pairs. Given thirty subjects, there are nine observed incidents for each pair of sets.

## **2nd Objective**

Each subject plays 25 games against the initial training phase opponents (i.e., opponent 4—OP4) and 25 games against the online trained opponents that were saved (i.e., two sets of games). We let each subject play another two sets against these opponents in different order. Half of the subjects play these four sets of games in the sequence OLL-OP4, OP4-OLL, whereas the other half play them in the sequence OP4-OLL, OLL-OP4, since we require minimization of any potential ordering effect. Each time a pair of sets (two pairs here) is finished, the player is asked whether the first set was more interesting than the second set of games.

Subjects are asked to list the criteria they used for their assessment of which set of games was more interesting.

## **STATISTICAL METHOD**

For this experiment, there are three null hypotheses formed.

- $H_0$ : The correlation between observed human judgment of interest and the computed interest value, as far as the different opponents are concerned, is a result of randomness.
- $H_1$ : Observed human judgment of interest does not correlate with the computed interest value, as far as the different opponents are concerned.
- $H_2$ : Observed human judgment of interest does not correlate with performance during play.

Given the interest metric (5) and two sets of games  $A$  and  $B$ , it can be determined that “game A is more (or less) interesting than game B.” In answer to the same question, a human subject can indicate that either

$I_A > I_B$  or  $I_A < I_B$ . In order to measure the degree of agreement between the human judgment of interest and the interest value given by (5), we calculate the correlation coefficients

$$c(\vec{z}) = \sum_{i=1}^N z_i / N, \quad (9)$$

where  $N$  is the number of incidents to correlate and

$$\vec{z} = \begin{cases} 1, & \text{if subject agrees with (5);} \\ -1, & \text{if subject disagrees with (5).} \end{cases} \quad (10)$$

The test statistic (9) is used to assess the truth of all three null hypotheses. However, for the null hypothesis  $H_2$ , the correlation coefficients  $c(\vec{z}')$  are computed where  $z'$  values are obtained from (11).

$$\vec{z}' = \begin{cases} 1, & \text{if subject chooses according to performance;} \\ -1, & \text{if subject does not choose according to performance.} \end{cases} \quad (11)$$

The distribution used for obtaining the correlation coefficient probabilities (p-values— $P(C \geq c)$ ) is the binomial. The observed effect is “highly significant” and “significant” if  $P(C \geq c) < 1\%$  and  $1\% < P(C \geq c) \leq 5\%$ , respectively.

For the design of the subjects’s self reports we follow the principles of comparative fun analysis (Read et al. 2002; Yannakakis et al. 2006). The endurability and expectations for the majority of subjects that played Pac-Man were very high, indicating that the game design used was successful. More specifically, all subjects were excited to play Pac-Man as soon as they were informed about the rules of the game (derived through a *Funometer* tool application [Read et al. 2002]) and the majority of subjects stressed that they would like to play the game again (derived through an Again-Again table [Read et al. 2002]). As previously mentioned, we use the 2-alternative forced choice (2-AFC) approach since it offers several advantages for a subjective entertainment capture. The 2-AFC comparative fun analysis minimizes the assumptions about people’s different notions of entertainment and provides data for a fair comparison among answers of different people.

## STATISTICAL ANALYSIS

As noted previously, this article concentrates on the characters’ behavioral aspect of interesting games. More specifically, it focuses on the opponent’s rather than the graphics’ or the sound’s impact on the player’s entertainment. Apart from the opponent, there are two additional factors that may affect the interest of a computer game, that are examined in this

section. These are the player-subject type (degree of *a priori* game liking) and the order of play.

## Opponent

Each entity in Table 3 represents a subject's answer to the question from the 1st objective, equivalent to "Is  $I_i > I_j?$ ," where  $i, j$ , are the row and column number, respectively. Given the interest values of the five opponents (see Table 2), "O and X" stand, respectively, for the subject's agreement and disagreement with this ranking (in other words, O and X characters are selected for visual purposes to symbolize the respective  $z$  values—see (10). As stressed before, given 30 subjects, there are nine incidents for each pair of opponent which are represented in a  $3 \times 3$  matrix. Rows within this matrix denote the type of subject that answered the specific question.

Table 4 presents the correlation coefficients and their respective  $P(C \geq c)$  values for each one of the ten combinations of opponent pairs ( $N = 18$ ) and in total ( $N = 180$ ). There is an obvious disagreement between the interest metric and the human's notion of interest in opponent pairs 1–2 and 3–4. Even though humans seem to agree with the interest metric in the pairs 1–3 and 1–4, the obtained p-values reveal statistically insignificant results. For the rest of the pairs, we experience statistically highly significant (i.e., 2–3, 2–5, 4–5) and significant (i.e., 1–5, 2–4, 3–5) matching to

**TABLE 3** Agreement Between the Subject's Judgment of Interest and the Interest Metric—O:  $z = 1$ , X :  $z = -1$

|              |            | Is $I_{Row} > I_{Column}?$ |       |       |       |       |
|--------------|------------|----------------------------|-------|-------|-------|-------|
|              |            | 1                          | 2     | 3     | 4     | 5     |
| Subject type |            |                            |       |       |       |       |
| 1            | Like       |                            | O O X | O O X | O O X | O O O |
|              | Neutral    |                            | O O X | O O O | O O X | O O X |
|              | Don't Like |                            | O O O | O O O | O O O | O X X |
| 2            | Like       | X X X                      |       | O O O | O O X | O O O |
|              | Neutral    | X X X                      |       | O O X | O O O | O O X |
|              | Don't Like | O X X                      |       | O O O | O O X | O O X |
| 3            | Like       | O X X                      | O O O |       | O X X | O O X |
|              | Neutral    | O O X                      | O O X |       | O X X | O O O |
|              | Don't Like | O X X                      | O O X |       | O O X | O O X |
| 4            | Like       | O O X                      | O O O | X X X |       | O O X |
|              | Neutral    | O O X                      | O O X | X X X |       | O O X |
|              | Don't Like | O X X                      | O O X | O X X |       | O O O |
| 5            | Like       | O O X                      | O O O | O O O | O O O |       |
|              | Neutral    | O O O                      | O O X | O O X | O O O |       |
|              | Don't Like | O O O                      | O O O | O O X | O O O |       |

**TABLE 4** Interest Metric – Subject Judgment Correlation Coefficients  $c$ ,  $P(C \geq c)$  Values, Order of Play Test Statistic  $z''$ , and  $P(Z \geq |z''|)$  Values for all Pairs of Opponents and in Total

| Pair  | $c$    | $P(C \geq c)$        | $z''$  | $P(Z \geq  z'' )$ |
|-------|--------|----------------------|--------|-------------------|
| 1–2   | −0.111 | 0.7596               | 0.2222 | 0.2403            |
| 1–3   | 0.3333 | 0.1189               | 0.3333 | 0.1189            |
| 1–4   | 0.3333 | 0.1189               | −0.222 | 0.2403            |
| 1–5   | 0.5555 | 0.0154               | −0.111 | 0.4072            |
| 2–3   | 0.6666 | 0.0037               | 0.1111 | 0.4072            |
| 2–4   | 0.5555 | 0.0154               | 0.1111 | 0.4072            |
| 2–5   | 0.7777 | 0.0006               | 0.0000 | 0.5927            |
| 3–4   | −0.444 | 0.9845               | −0.111 | 0.4072            |
| 3–5   | 0.5555 | 0.0154               | −0.333 | 0.1189            |
| 4–5   | 0.6666 | 0.0037               | 0.2222 | 0.2403            |
| Total | 0.3888 | $1.31 \cdot 10^{-7}$ | 0.0222 | 0.4818            |

observed human judgment. Finally, the total agreement correlation coefficient ( $c = 0.3888$ ), as well as its p-value ( $P(C \geq c) = 1.31 \cdot 10^{-7}$ ), demonstrate a statistically highly significant effect that rules out the null hypothesis  $H_1$ . Thus, it appears that the observed human judgment of interest correlates with the computed interest value, as far as the different opponents are concerned. Moreover, the obtained p-values presented in Table 4 illustrate that the sample size of 30 subjects is adequate to produce statistically significant observed effects.

### ***Opponent 1***

Further investigation of the interest value generated by opponent 1 showed high dependence on the player type. More specifically, when opponent 1 plays against the CB *Pac-Man* and the RB *Pac-Man*, it generates interest, which is respectively statistically not different and significantly higher than the interest generated by opponent 2. Opponent 1 constitutes a particular case since no such change in the opponent ranking (i.e., ranked by interest) occurs for any other of the four remaining opponents.

Given the ranking instability of opponent 1, we recalculate the  $z$  values as if (1)  $I_1 > I_2$  and (2)  $I_1 = I_2$  and proceed. In the former case, the  $z$  values of the [1–2] pair swap their sign and the obtained p-values for this pair and in total are 0.4072 and  $2.57 \cdot 10^{-8}$ , respectively. For the latter case, the  $z$  values of the [1–2] pair are not taken into consideration and the two first (triplets of) rows and columns of Table 3 are merged into one by adding up their  $z$  values. The obtained p-values for the merged pairs and in total are presented in Table 5. For both cases, changes in the opponent 1 ranking increase the significance of the observed effects.

**TABLE 5** Interest Metric: Subject Judgment Correlation Coefficients  $c$  and  $P(C \geq c)$  Values When  $I_1 = I_2$  Is Assumed

| Pairs   | $c$    | $P(C \geq c)$        |
|---------|--------|----------------------|
| (1,2)-3 | 0.5000 | 0.0019               |
| (1,2)-4 | 0.4444 | 0.0056               |
| (1,2)-5 | 0.6667 | $3.5 \cdot 10^{-5}$  |
| Total   | 0.4444 | $1.17 \cdot 10^{-8}$ |

## Order of Play

In order to check whether the order of playing Pac-Man games affects the human judgment of interest, we hypothesize that there is no order effect and proceed as follows. For each pair of opponents, that a subject played in both orders, we count (a) the times  $K$  that the subject agrees with the interest value only in the first pair played and (b) the times  $J$  that the subject agrees with the interest value only in the latter pair played. In the case where the subject agrees or disagrees with the interest value in both pairs played, we take no action. To this end, we compute the  $z''$  value (12) for each pair of opponents ( $N = 9$ ) and in total ( $N = 90$ ).

$$z''(K, J) = (K - J)/N \quad (12)$$

The greater the absolute value of  $z''(K, J)$ , the more the order of play tends to affect the subjects' judgment of interest. This value defines the test statistic used to assess the truth of the hypothesis that there is no order effect. The obtained  $z''$  value is trinomially distributed.

As seen from Table 4, there are no statistically significant effects in any pair of opponents or in total. Therefore, the null hypothesis is not rejected and it seems that the order of play does not affect the human judgement of interest.

## Opponent 1

The order of play is not affected by the particular behavior of opponent 1 either. That is, if  $I_1 > I_2$ , there is no difference in the obtained  $P(Z \geq |z''|)$  values and, if  $I_1 = I_2$ , there is no statistically significant effects in any pair of opponents or in total (i.e.,  $P(Z \geq |z''|) = 0.5312$ ).

## Subject Type

In this section, we present how the subject's type, which corresponds to the subject's "liking of the Pac-Man game," correlates with the subject's judgment of interest. To this end, we compute the correlation coefficients  $c$  and their respective probabilities  $P(C \geq c)$  for each subject type (60 incidents for each type).

**TABLE 6** Interest Metric – Subject Judgment Correlation  
 Coefficients  $c$ ,  $P(C \geq c)$  Values of the Three Types of Subject and  
 Correlation Variance ( $\sigma_c^2$ ) Over the 10 Subjects of Each Type

| Subject type | $c$    | $(\sigma_c^2)$ | $P(C \geq c)$        |
|--------------|--------|----------------|----------------------|
| Like         | 0.4000 | 0.0691         | 0.0013               |
| Neutral      | 0.6333 | 0.1234         | 0.0067               |
| Don't like   | 0.4333 | 0.1493         | 0.0005               |
| Total        | 0.3888 | 0.1079         | $1.31 \cdot 10^{-7}$ |

As seen from Table 6, all three types of subject's observed judgment of interest collectively demonstrate a highly significant agreement ( $P < 1\%$ ) with the interest metric. However, it appears that there is no significant difference between the three types and, therefore, no secure conclusions about the subject's type effect on its notion of interest can be arisen.

### *Opponent 1*

By following the procedure described previously, for the particular case of opponent 1, we also come up with highly significant values for all three subject types and no significant difference between them for both cases of  $I_1 > I_2$  and  $I_1 = I_2$ .

### **Online Learning**

In this section, we analyze the observed effects from the on-line learning experiment (Part C) presented previously. In Part C, subjects play 2 sets of 50 games in total. Interest values calculated (bootstrapping procedure with  $N = 25$ ) and presented in Table 7 show that, in 18 out of 30 cases, the human player managed to produce more interesting games by the use of the online learning procedure. However, it is not clear whether OLL used in humans cause the interest value to proliferate. Thus, it seems that 50 OLL games (testing period in Part A) are not adequate for the OLL mechanism to cause a significant difference in the interest value (see Table 7).

Choosing an online learning period (or else testing period) of 50 games is an empirical way of balancing efficiency and experimental time. The duration of the testing period lasted 20 minutes on average, whereas the whole experiment exceeded 65 minutes in many cases, which is a great amount of time for a human to be constantly concentrated. Fixed strategy *Pac-Man* player results showed that more online learning games are required for the interest value to change significantly, which appears to be the case for human players as well.

By calculating the correlation coefficient (9) between the computed interest values (presented in Table 7) and the human judgment of interest

**TABLE 7** Interest  $I$  and Confidence Interval ( $I_u, I_l$ ) Values Against All 30 Human Players Ranked by Subject Type. i.e. 1–10: Like, 11–20: Neutral, 21–30: Don't Like

| Subject      | OLL   |       |       | No OLL |       |       |
|--------------|-------|-------|-------|--------|-------|-------|
|              | $I_u$ | $I_l$ | $I$   | $I_u$  | $I_l$ | $I$   |
| 1            | 0.721 | 0.575 | 0.671 | 0.745  | 0.393 | 0.630 |
| 2            | 0.753 | 0.588 | 0.669 | 0.767  | 0.593 | 0.703 |
| 3            | 0.733 | 0.614 | 0.669 | 0.755  | 0.607 | 0.694 |
| 4            | 0.805 | 0.672 | 0.735 | 0.792  | 0.520 | 0.677 |
| 5            | 0.802 | 0.644 | 0.711 | 0.720  | 0.582 | 0.665 |
| 6            | 0.763 | 0.598 | 0.676 | 0.733  | 0.531 | 0.647 |
| 7            | 0.725 | 0.638 | 0.689 | 0.698  | 0.559 | 0.644 |
| 8            | 0.751 | 0.566 | 0.673 | 0.804  | 0.603 | 0.720 |
| 9            | 0.746 | 0.568 | 0.681 | 0.751  | 0.630 | 0.698 |
| 10           | 0.780 | 0.531 | 0.670 | 0.780  | 0.616 | 0.715 |
| 11           | 0.692 | 0.469 | 0.619 | 0.750  | 0.576 | 0.695 |
| 12           | 0.802 | 0.678 | 0.748 | 0.865  | 0.700 | 0.778 |
| 13           | 0.806 | 0.530 | 0.662 | 0.716  | 0.532 | 0.638 |
| 14           | 0.799 | 0.589 | 0.715 | 0.805  | 0.678 | 0.738 |
| 15           | 0.776 | 0.636 | 0.707 | 0.782  | 0.656 | 0.706 |
| 16           | 0.812 | 0.658 | 0.749 | 0.806  | 0.689 | 0.745 |
| 17           | 0.784 | 0.601 | 0.706 | 0.743  | 0.609 | 0.679 |
| 18           | 0.796 | 0.595 | 0.708 | 0.740  | 0.567 | 0.655 |
| 19           | 0.780 | 0.612 | 0.702 | 0.718  | 0.626 | 0.670 |
| 20           | 0.749 | 0.666 | 0.717 | 0.759  | 0.646 | 0.716 |
| 21           | 0.753 | 0.625 | 0.684 | 0.757  | 0.659 | 0.706 |
| 22           | 0.790 | 0.660 | 0.728 | 0.831  | 0.625 | 0.733 |
| 23           | 0.774 | 0.640 | 0.709 | 0.762  | 0.663 | 0.700 |
| 24           | 0.752 | 0.599 | 0.668 | 0.754  | 0.612 | 0.681 |
| 25           | 0.741 | 0.635 | 0.696 | 0.705  | 0.589 | 0.660 |
| 26           | 0.825 | 0.697 | 0.770 | 0.781  | 0.681 | 0.728 |
| 27           | 0.799 | 0.622 | 0.732 | 0.782  | 0.640 | 0.724 |
| 28           | 0.786 | 0.630 | 0.719 | 0.755  | 0.570 | 0.693 |
| 29           | 0.745 | 0.607 | 0.690 | 0.748  | 0.606 | 0.705 |
| 30           | 0.793 | 0.673 | 0.738 | 0.782  | 0.591 | 0.678 |
| $E\{\cdot\}$ | 0.771 | 0.614 | 0.700 | 0.763  | 0.605 | 0.694 |

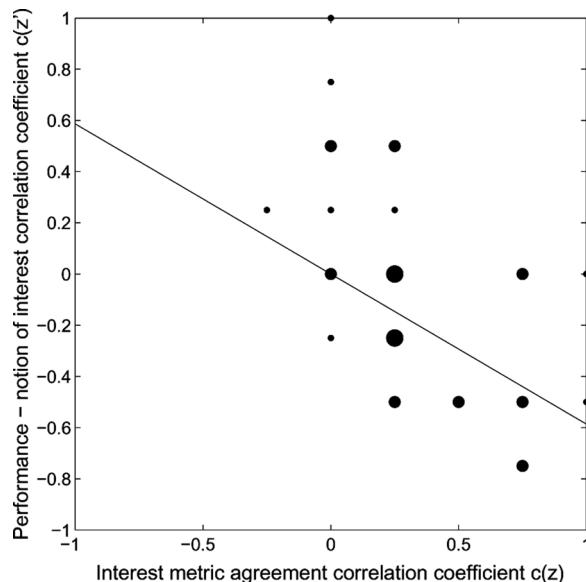
obtained the first question of our second objective, we get a value of  $c = 0.1666$ , with corresponding probability of  $P(C \geq c) = 0.1225$  for  $N = 60$ . This does not constitute a statistically significant effect and suggests that humans were not able to tell the difference between opponent 4 and the opponents trained online at the end of the testing period.

### Performance Factor

As noted before, each subject plays eight pairs of sets of games in total during this experiment (six in Part B and two in Part C), and each set is assigned a score that corresponds to the performance of the subject. More

specifically, the score is directly proportional to the number of pellets eaten by the player. Given the subjects' scores and the reported interest judgement, the  $z'$  values are computed as follows. If the subject chooses the set of games with the higher score obtained as being more interesting, then the  $z'$  value is 1. Accordingly, the  $z'$  value is  $-1$  if the subject chooses the set of games with the lower score obtained as being more interesting. By computing (9) for all 30 subjects ( $N = 8 \cdot 30 = 240$ ), we get  $c(z') = -0.05$  and  $P(C \geq -0.05) = 0.7994$ , which constitutes the effect as statistically not significant. Therefore, the null hypothesis  $H_2$  is not rejected and it seems that observed human judgment of interest does not correlate with performance during play.

However, before abandoning the hypothesis of the performance impact on human judgment totally, we attempt to draw the relation between the two from another perspective. Figure 5 illustrates a scatter plot of the correlation coefficients between the performance and the subject's judgement of interest against the correlation coefficients between the interest metric and the subject's judgment of interest for each subject. In addition, the line of the statistical correlation ( $f(x) = \text{cor}(c(\vec{z}), c(z')) \cdot x$ , where  $\text{cor}(c(\vec{z}), c(z')) = \text{cov}(c(\vec{z}), c(z'))/\sigma_{c(\vec{z})}\sigma_{c(z')} = -0.5864$ ) between the two samples of data is plotted. If we examine Figure 5 in detail as well as the reported interest criteria in the last question of the survey, there seems to be a classification of the subjects into three groups. These are:



**FIGURE 5** Scatter plot of  $c(z)$  and  $c(z')$  values for each subject and their statistical correlation' line. The circular marker' radius is increased in respect to the number of occurrences (i.e., 1, 2 or 3).

- Subjects that judge interest according to their performance ( $c(z') \geq 0.5$ ), size: 6 out of 30 subjects. As far as their agreement with the interest metric is concerned, their observed judgment portrays a rather random behavior ( $0.0 \leq c(z) \leq 0.25$ ). The reported interest criteria in the last question of the survey are explicit. Randomness and scoring performance are the major criteria in selecting the most interesting set between two.
- Subjects that do not judge interest according to their performance ( $c(z') \leq 0.0$ ) and whose interest judgment correlates with the interest metric ( $c(z) \geq 0.5$ ), size: 12 out of 30 subjects. This group's reported interest criteria are focused on the opponent's contribution to the player's satisfaction.
- Subjects that do not judge interest according to their performance ( $-0.5 < c(z') < 0.5$ ) and whose interest judgment does not correlate with the interest metric ( $c(z) < 0.5$ ), size: 12 out of 30 subjects. Subjects of this group concentrate on a variety of Pac-Man aspects different or implicitly syngeneic to the *Ghosts'* behavior, as acquired from the reported interest criteria. These aspects include performance, game control ability, graphics, difficulty, and duration of game.

The computed statistical correlation value and Figure 5 provide evidence that human judgment of interest, that agrees with the interest metric, is not correlated with the human judgment of interest based on performance. In other words, it seems that subjects agreeing with the interest metric do not judge interest by their performance. Or else, subjects disagreeing with the interest metric seem to judge interest by their score and/or other criteria such as game controls and graphics.

### ***Opponent 1***

By assuming that  $I_1 > I_2^2$ , we reveal a slightly higher statistical correlation value  $\text{cor}(c(\vec{z}), c(z')) = -0.5341$ , but conceptually the same effects and subject classification groups as the above-mentioned.

## **CONCLUSIONS**

Predator strategies in predator/prey computer games are still nowadays based on simple rules, which make the game pretty predictable and, therefore, somewhat uninteresting (by the time the player gains more experience and playing skills) (Woodcock 2001; Rabin 2002). A computer game becomes enjoyable primarily when there is a richer online interaction between the player and its opponents who demonstrate interesting behaviors (Yannakakis and Hallam 2004a; 2005c). Machine learning techniques applied online (Stanley et al. 2005) can generate behaviors that give the

illusion of intelligence, which is an important criterion for the human player's perceived entertainment. On top of that, playing against cooperative opponents makes the game more realistic and appealing to the human eye (Yannakakis and Hallam 2005b).

Given some objective criteria for defining interest in predator/prey games, we introduced a generic method for measuring interest in such games. The  $I$  metric presented in this article captures the concept of interest objectively and it is dependent on the player-game opponent interaction. We saw that by using the proposed online learning mechanism on the Pac-Man game, maximization of the individual simple distance measure (see (7)) coincides with maximization of the game's interest. Apart from being fairly robust, the proposed approach demonstrated high and fast adaptability to changing types of player (i.e., playing strategies). Results obtained against fixed strategy *Pac-Man* players showed that such a mechanism could be able to produce interesting interactive opponents (i.e., games) against dynamic human playing strategies.

By testing the game against humans, we managed to confirm our hypothesis that the interest value computed by (5) is consistent with the judgment of human players. In fact, human player's notion of interest of the Pac-Man game correlate highly with the captured interest value. However, there are instances where humans' reported notion of interest does not match the respective calculated  $I$  value of the game. Since the proposed interest metric was designed and evaluated on computer-controlled *Pac-Man* players, the reported mismatches confirm the fact that a human playing behavior differs from a computer-controlled designed player. In addition, it is revealed that both the subject type (i.e., experience/likeness with the game) and the order of playing the game do not affect their judgment. Moreover, given each subject's score, it was demonstrated that humans agreeing with the interest metric do not judge interest by their performance. Or else, humans disagreeing with the interest metric judge interest by their score or based on other personal criteria like game control and graphics.

The main assumption drawn for the interest metric proposed is that players overall have a basic level of gaming skills for the test-bed game. In that sense, the computer-guided players used are models of some well-behaved, average skill players based on similar motion patterns that do not leave much space for significant differences in their best generated interest values. Human players that tested Pac-Man cross-validate this assumption since their generated interest values against the same opponent were not significantly different from each other.

As far as online learning against human players is concerned, results show that more online learning games are required for the interest value to change significantly and for humans to notice some sort of change in

the interest of the game. This is a function of the way that human-game interaction is used to train the opponents. Given more computing power, it may be possible to use the data provided by human-game interaction more efficiently and therefore achieve significant change in interest in fewer games. For instance, thousands of mutants could be evaluated in parallel over longer periods—which would provide better behavior estimates—and moreover the frequency of evolutionary iterations could be increased. Using this approach, we could accelerate the learning where appropriate and minimize the probability of unwanted, unrealistic, non-intelligent generated behaviors due to mutation. Clearly, a single unrealistic emerged AI behavior is sufficient to impair the “intelligent” image any adaptive approach is attempting to present and furthermore to diminish the satisfaction of the player (Champandard 2004; Funge 2004).

## Discussion

As a novel direction in AI in computer games, cooperative opponents that learn in real time for optimizing the entertainment value of the game constitutes this work’s proposed step for future game development. Showing that such a learning mechanism maintains high levels of player satisfaction makes this approach appealing for application to the vast majority of game genres where online learning and opponent cooperation are, until nowadays, deliberately absent or optional. Here we discuss the potential of the methodology in other genres of multi-opponent games where the complication of the opponents’ tasks may differ. More specifically, we analyze the extensibility of the interest metric proposed, the online evolutionary learning mechanism and the neuro-controller used.

1. *Interest Metric* As already mentioned, the criteria of challenge and behavior’s diversity may be effectively applied for measuring the real-time entertainment value of any genre of games. Spatial diversity may in a sense also contribute to the interest value of specific genres (e.g., team-sport, real-time strategy, and first-person shooter games). As long as game developers can determine and extract the features (e.g., through online questionnaires) of the opponent behavior that generate excitement for the player, a mathematical formula can be designed in order to collectively represent them.
2. *Learning Methodology* The proposed online evolutionary learning method may also be successfully applied to any game during active real-time player-opponent interactions. Extracted features of this interaction may be used in order to estimate the fitness of the involved opponents according to their tasks. The replacement of the worst-fit opponent(s) method may be applied in frequent game periods to enhance the

group's fitness. See also Stanley et al. (2005) for a successful application of this method in the NERO game.

Artificial evolution can explore complex search spaces efficiently and, when combined with NNs, it can demonstrate fast adaptability to dynamic and changing environments. Therefore neuro-evolution is recommended for learning in real time. However, convergence time and unpredictability of the emergent behaviors constitute the disadvantages of the methodology, which can be dealt with by careful design of the learning mechanism. The tradeoff between opponent behavior stability and speed of learning online when using neuro-evolution was addressed in this article. Both the breeding of offspring and the variance of the Gaussian mutation used are inversely proportional to the cell visit entropy. This GA scheme minimizes unpredictability and allows for rapid genotype alterations when the interest value of the game is high and low, respectively.

In addition to a careful GA design, player modeling techniques are able to decrease convergence time down to realistic periods of time (i.e., tens of games) and furthermore proliferate the efficiency and justifiability of learning in real time (Yannakakis and Maragondiakis 2005).

3. *Controller* Artificial neural networks serve successfully the adaptability requirements for predator/prey reactive games in real-time. However, as the complexity of the opponents' tasks increases, there might be a need for more sophisticated structures of distributed representation. Memory of previous behaviors learned through the player-opponent interaction may very well be essential when a combination of various tasks is required. Recurrent NNs or augmented NN topologies with hidden states (Stanley and Miikulainen 2002) may be more appropriate when the opponents' tasks proliferate. Moreover, a hierarchy design of neuro-controllers that serve different opponent tasks could also provide the online learning mechanism with more flexibility and faster adaptability. Decision trees, adaptive scripts (Spronck et al. 2001), or classifier systems (Champandard 2004) could also host adaptive behaviors in real time successfully.

## REFERENCES

- Ackley, D. H. and M. L. Littman. 1992. Interactions between learning and evolution. In: *Artificial Life II*, eds. C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, 478–507. Reading, MA: Addison-Wesley.
- Andrade, G., G. Ramalho, H. Santana, and V. Corruble. 2005. Extending reinforcement learning to provide dynamic game balancing. In: *Proceedings of the Workshop on Reasoning, Representation, and Learning in Computer Games, 19th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 7–12.
- Champandard, A. J. 2004. *AI Game Development*. New Riders Publishing.
- Cole, N., S. J. Louis, and C. Miles. 2004. Using a genetic algorithm to tune first-person shooter bots. In: *Proceedings of the 2004 Congress on Evolutionary Computation*, pages 139–145.

- Csikszentmihalyi, M. 1990. *Flow: The Psychology of Optimal Experience*. New York: Harper & Row.
- Demasi, P. and A. J. de O. Cruz. 2002. On-line coevolution for action games. In: *Proceedings of the 3rd International Conference on Intelligent Games and Simulation (GAME-ON)*, pages 113–120.
- Fogel, D. B. 1993. Using evolutionary programming to construct neural networks that are capable of playing tic-tac-toe. In: *Proceedings of the IEEE International Conference on Neural Networks*, page 875–880, San Francisco, CA, USA: IEEE Press.
- Fogel, D. B. 2002. *Blondie 24: Playing at the Edge of AI*. Morgan Kaufmann.
- Fogel, D. B., T. J. Hays, and D. R. Johnson. 2004. A platform for evolving characters in competitive games. In: *Proceedings of the Congress on Evolutionary Computation (CEC-04)*, pages 1420–1426, June 2004.
- Funge, J. D. 2004. *Artificial Intelligence for Computer Games*. Wellesley, MA: A. K. Peters Ltd.
- Gallagher, M. and A. Ryan. 2003. Learning to play Pac-man: An evolutionary, rule-based approach. In: *Proceedings of the Congress on Evolutionary Computation (CEC)*, pages 2462–2469.
- Graepel, T., R. Herbrich, and J. Gold. 2004. Learning to fight. In: *Proceedings of the International Conference on Computer Games: Artificial Intelligence, Design and Education*, pages 193–200, Reading, UK.
- Haynes, T. and S. Sen. 1995. Evolving behavioral strategies in predators and prey. In: *IJCAI-95 Workshop on Adaptation and Learning in Multiagent Systems*, pages 32–37, Montreal, Quebec, Canada: Morgan Kaufmann.
- Hunicke, R. and V. Chapman. 2004. AI for dynamic difficulty adjustment in games. In: *Proceedings of the Challenges in Game AI Workshop, 19th Nineteenth National Conference on Artificial Intelligence (AAAI'04)*, pages 91–96.
- Iida, H., N. Takeshita, and J. Yoshimura. 2003. A metric for entertainment of boardgames: Its implication for evolution of chess variants. In: *IWEC2002 Proceedings*, pages 65–72, Kluwer.
- Isla, D. and B. Blumberg. 2002. New challenges for character-based AI for games. In: *Proceedings of the AAAI Spring Symposium on AI and Interactive Entertainment*, pages 41–45, Stanford, CA, USA: AAAI Press.
- Johnson, S. 2004. *Adaptive AI*. Hingham, MA: Charles River Media.
- Kaiser, S. and T. Wehrle. 1996. Situated emotional problem solving in interactive computer games. In: *Proceedings of the VIXth Conference of the International Society for Research on Emotions*, pages 276–280, ISRE Publications.
- Kaiser, S., T. Wehrle, and S. Schmidt. 1998. Emotional episodes, facial expressions, and reported feelings in human computer interactions. In: *Proceedings of the Xth Conference of the International Society for Research on Emotions*, pages 82–86, ISRE Publications.
- Khoo, A. and R. Zubeck. 2002. Applying inexpensive techniques to computer games. *IEEE Intelligent Systems*, 17(4): 48–53.
- Koster, R. 2005. *A Theory of Fun for Game Design*. Scottsdale, AZ: Paraglyph Press.
- Koza, J. 1992. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press.
- Laird, J. E. 2002. Research in human-level AI using computer games. *Communications of the ACM* 3(8): 32–35.
- Laird, J. E. and M. van Lent. 2000. Human-level AI's killer application: Interactive computer games. In: *Proceedings of the Seventh National Conference on Artificial Intelligence (AAAI)*, pages 1171–1178, Saint Paul, MN.
- Lazzaro, N. 2004. *Why We Play Games: Four Keys to More Emotion Without Story*. XEO Design Inc., Technical Report.
- Lucas, S. 2005. Evolving a neural network location evaluator to play Ms. Pac-Man. In: *Proceedings of the IEEE Symposium on Computational Intelligence and Games*, pages 203–210, Cambridge, MA: Colchester, UK.
- Luke, S. and L. Spector. 1996. Evolving teamwork and coordination with genetic programming. In: *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 150–156, Palo Alto, California, USA: MIT Press.
- Malone, T. W. 1981. What makes computer games fun. *Byte* 6:258–277.
- McQuiggan, S., S. Lee, and J. Lester. 2006. Predicting user physiological response for interactive environments: An inductive approach. In: *Proceedings of the 2nd Artificial Intelligence for Interactive Digital Entertainment Conference*, pages 60–65.

- Miller, G. and D. Cliff. 1994. Protean behavior in dynamic games: Arguments for the co-evolution of pursuit-evasion tactics. In: *From Animals to Animats 3: Proceedings of the Third International Conference on Simulation of Adaptive Behavior (SAB-94)*, pages 411–420, MIT Press.
- Ponsen, M. and P. Spronck. 2004. “Improving adaptive game AI with evolutionary learning. In: *Proceedings of the International Conference on Computer Games: Artificial Intelligence, Design and Education*, pages 389–396, Reading, UK.
- Rabin, S. 2002. *AI Game Programming Wisdom*.: Charles River Media, Inc.
- Rani, P., N. Sarkar, and C. Liu. 2005. Maintaining optimal challenge in computer games through real-time physiological feedback. In *Proceedings of the 11th International Conference on Human Computer Interaction*, pages 184–192.
- Read, J., S. MacFarlane, and C. Cassey. 2002. Endurability, engagement and expectations. In *Proceedings of International Conference for Interaction Design and Children*, ACM Press.
- Richards, N., D. E. Moriarty, and R. Miikkulainen. 1998. Evolving neural networks to play go. *Appl. Intell.* 8(1):85–96.
- Rosca, J. 1996. Generality versus size in genetic programming. In *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 381–387, Palo Alto, California, USA: MIT Press.
- Spronck, P., I. Sprinkhuizen-Kuyper, and E. Postma. 2004. Difficulty scaling of game AI. In: *Proceedings of the 5th International Conference on Intelligent Games and Simulation (GAME-ON 2004)*, pages 33–37.
- Stanley, K., Bryant, B., and R. Miikkulainen. 2005. Real-time evolution in the NERO video game. In: *Proceedings of the IEEE Symposium on Computational Intelligence and Games*, pages 182–189, Colchester, UK.
- Stanley, K. and R. Miikkulainen. 2002. Evolving neural networks through augmenting topologies: *Evolutionary Computation* 10(2):99–127.
- Sweetser, P. and P. Wyeth. 2005. GameFlow: A model for evaluating player enjoyment in games. *ACM Computers in Entertainment* 3(3).
- Taylor, T. 2000. “Artificial life techniques for generating controllers for physically modelled characters.” In: *Proceedings of the First International Conference on Intelligent Games and Simulation (GAME-ON 2000)*, ■.
- Tesauro, G. 2002. Programming backgammon using self-teaching neural nets. *Artificial Intelligence* 134:181–199.
- Thurau, C., C. Bauckhage, and G. Sagerer. 2004. Learning human-like movement behavior for computer games, In: *From Animals to Animats 8: Proceedings of the 8th International Conference on Simulation of Adaptive Behavior (SAB-04)*, pages 315–323, Santa Monica, CA: The MIT Press.
- Verma, M. A. and P. W. McOwan. 2005. An adaptive methodology for synthesising mobile phone games using genetic algorithms. In: *Proceedings of the Congress on Evolutionary Computation (CEC-05)*, pages 528–535, Edinburgh, UK.
- Weizenbaum, J. 1966. ELIZA—a computer program for the study of natural language communications between men and machines. *Communications of the Association for Computing Machinery* 9:36–45.
- Wikipedia, the Free Encyclopedia. 2005. Pac-Man, [Online]. Available: <http://en.wikipedia.org/wiki/Pac-man>
- Woodcock, S. 2001. *Game AI: The State of the Industry 2000–2001: It's not Just Art, It's Engineering*.
- Yannakakis, G. N. 2005. *AI in Computer Games: Generating Interesting Interactive Opponents by the use of Evolutionary Computation*. Ph.D. thesis, University of Edinburgh.
- Yannakakis, G. N. and J. Hallam. 2004a. “Evolving opponents for interesting interactive computer games.” In: *From Animals to Animats 8: Proceedings of the 8th International Conference on Simulation of Adaptive Behavior (SAB-04)*, pages 499–508, Santa Monica, CA, USA: The MIT Press.
- Yannakakis, G. N. and J. Hallam. 2004b. *Interactive opponents generate interesting games*. In: *Proceedings of the International Conference on Computer Games: Artificial Intelligence, Design and Education*, pages 240–247, Reading, UK.
- Yannakakis, G. N. and J. Hallam. 2005a. A generic approach for generating interesting interactive Pac-man opponents. In: *Proceedings of the IEEE Symposium on Computational Intelligence and Games*, pages 94–101, Colchester, UK.
- Yannakakis, G. N. and J. Hallam. 2005b. A generic approach for obtaining higher entertainment in predator/prey computer games. *Journal of Game Development* 1(3):23–50.

- Yannakakis, G. N. and J. Hallam. 2005c. A scheme for creating digital entertainment with substance. In: *Proceedings of the Workshop on Reasoning, Representation, and Learning in Computer Games, 19th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 119–124.
- Yannakakis, G. N. and J. Hallam. 2006. Towards capturing and enhancing entertainment in computer games. In: *Proceedings of the 4th Hellenic Conference on Artificial Intelligence, Lecture Notes in Artificial Intelligence*, vol. 3955, pages 432–442 Heraklion, Greece: Springer-Verlag.
- Yannakakis, G. N. and M. Maragoudakis. 2005. Player modeling impact on player's entertainment in computer games. In *Proceedings of the 10th International Conference on User Modeling; Lecture Notes in Computer Science*, vol. 3538, pages 74–78. Edinburgh: Springer-Verlag.
- Yannakakis, G. N., J. Hallam, and H. H. Lund. 2006. Capturing entertainment through heart-rate dynamics in the playware playground. In *Proceedings of the 5th International Conference on Entertainment Computing, Lecture Notes in Computer Science*, vol. 4161, pages 314–317 Cambridge, UK: Springer-Verlag.
- Yannakakis, G. N., J. Hallam, and H. H. Lund. 2006. Comparative fun analysis in the innovative playware game platform. In *Proceedings of the 1st World Conference for Fun'n Games*, pages 64–70, Preston, UK.
- Yannakakis, G. N., J. Levine, and J. Hallam. 2004. An evolutionary approach for interactive computer games. In *Proceedings of the Congress on Evolutionary Computation (CEC-04)*, pages 986–993.
- Yao, X. 1999. Evolving artificial neural networks. *Proceedings of the IEEE* 87(9):1423–1447.

## NOTES

1. Further details of this strategy are presented in Yannakakis and Hallam (2004).
2. The case of  $I_1 = I_2$  is not investigated since the 1–2 pair is not taken into consideration and  $z'$  values cannot be computed for subjects that played that particular pair of sets.

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/221157583>

# Modeling enjoyment preference from physiological responses in a car racing game

Conference Paper · August 2010

DOI: 10.1109/ITW.2010.5593337 · Source: DBLP

---

CITATIONS  
61

READS  
110

---

4 authors:



Simone Tognetti  
Empatica S.r.l.  
17 PUBLICATIONS 281 CITATIONS

[SEE PROFILE](#)



Maurizio Garbarino  
Politecnico di Milano  
9 PUBLICATIONS 248 CITATIONS

[SEE PROFILE](#)



Andrea Bonarini  
Politecnico di Milano  
232 PUBLICATIONS 2,331 CITATIONS

[SEE PROFILE](#)



Matteo Matteucci  
Politecnico di Milano  
274 PUBLICATIONS 3,249 CITATIONS

[SEE PROFILE](#)

Some of the authors of this publication are also working on these related projects:



Krog - Kinect RObot for Gaming - Polisocial [View project](#)



Incremental Dense 3D Reconstruction [View project](#)

# Modeling enjoyment preference from physiological responses in a car racing game

Simone Tognetti, Maurizio Garbarino, Andrea Bonarini, Matteo Matteucci

**Abstract**— We propose a framework to estimate player enjoyment preference from physiological signals. This can produce objective measures that could be used to adapt dynamically a game to maintain the player in an optimal status of enjoyment. We present a case study on The Open Racing Car Simulator (TORCS) video game. In particular, we focus both on the experimental protocol, which we designed with special attention to produce physiological responses related to the game experience only, and on signal analysis, which produces a simple and general model good enough to estimate player enjoyment preference in real applications.

## I. INTRODUCTION

“A good game is the one you like to play”. With this motto in her mind, the digital game designer conceives rules and structure of a game to maximize the level of enjoyment for the target audience. Realistic ambiance and characters, intelligent and reactive opponents with a human-like behavior can provide enjoyable game experience, however this cannot be assumed a priori. Some evaluation can be done by considering performance parameters, often assuming that a better performance is related to higher enjoyment of the game experience; but, different players may react differently to game features, and enjoyment has to be considered as a personal experience. According to the affective computing, we can assume that physiological response can be related to enjoyment [1], and that it can be taken as an objective measure of it. Thus, we present in this paper the application of a methodological framework for the estimate of preference among game experiences, from the physiological state of the player, in the car racing video game scenario implemented by TORCS – The Open Racing Car Simulator [2].

With the purpose of comparing and validating the proposed game scenario with respect to recent literature, we have carried out a correlation analysis between physiological data and subject preference during different variants of the game. Results show that features derived from some of the physiological signals (e.g., Galvanic Skin Response (GSR), Blood Volume Pulse (BVP) and Respiration (RESP)) have a high correlation with player reported preferences. On the other hand, as we expected from other studies in literature, only some physiological features show relevant high discriminating power. Therefore, signals such as Heart Rate (HR) or temperature are not really suitable as emotional input because of their poor correlation with user preference. Supported by these findings, we have estimated a linear

model based on physiological signals able to predict the subject enjoyment preference during the game. Our approach focuses on the differential comparison of preference between game situations and the resulting model can be used, in a future experiment, to modify at runtime the game experience accordingly to the predicted user preference to optimize user enjoyment.

The ground truth about subject preference has been evaluated by questionnaire analysis. The players are asked to express a preference between two variants of the video game. This approach of eliciting emotion is named comparative affect analysis and it was first introduced by Yannakakis and Hallam [3], [4]. Affective models are then derived using preference learning techniques [5], [6] matching user reported preferences and features from physiological signals measured during the game session. We assume that physiological responses are not task dependent since the level of physical activity required for the interaction with our game is constant during the session. As presented by Yannakakis [7], the preference model can be learned using different computational methods. In this paper, we propose a different linear model obtained with Linear Discriminant Analysis (LDA) [8], which shows performance analogous to other models presented in literature [9], but lower complexity and lower computation demand.

In the next sections, we first give an insight into the state of the art, then we introduce the experimental protocol designed for our experiments, finally we present the methods we adopted for preference modeling and the results obtained.

### A. State of the art

With the purpose of determining criteria that contribute to player satisfaction, Yannakakis and Hallam [10] proposed two techniques for modeling player satisfaction in real-time. They assume that player-opponent interaction primarily contributes to the entertainment in a computer game. Therefore, metrics based on qualitative considerations of what is enjoyable from in-game performances (e.g., time before the player loses a life) have been considered as indicators of the level of interest.

Another approach consists in modeling the entertainment by following the theoretical principles described by Malone [11], and concepts related to the Theory of Flow [12]. Qualitative factors such as challenge, fantasy and curiosity are the ones that, according to them, mostly account for player entertainment. Quantitative measures for challenge, curiosity and flow state can be derived from an empirical analysis of player responses to game mechanisms. All the

Politecnico di Milano IIT Unit, Dip. di Elettronica e Informazione, via Ponzio 34/5, 20133 Milano, Italy. E-mail: {tognetti,garbarino,bonarini,matteucci}@elet.polimi.it.

metrics evaluated in the mentioned papers are specific for a given class of games (e.g., predator/prey, racing, football), therefore a general model can not be determined with these approaches. Moreover, player responses can be affected by factors such as user experience and motion or perception skills, which can be loosely related to enjoyment, while physiological responses seem to be better indicators.

Under a different perspective, emotion has been investigated in the past by many researchers, including philosophers, psychologists, sociologists, psychophysicists, and engineers. Results from psychophysiological studies [1], [13] indicated that relationship between the stimuli presented to a person and observed physiological reactions may exist. Grounded on these findings, people working on Affective Computing aimed to design human-machine interfaces, with emotion recognition abilities, for real life applications [14], [15], [16]. Recently, this research line has been extended to video games in which experiments can be performed with a good trade-off between reality and control.

In Mandryk et al. [17], statistically significant correlation has been claimed between GSR and reported fun (from questionnaire) in adults playing video games. Other physiological signals such as jaw electromyography, electrocardiography and respiration have been analyzed, but they resulted not to be correlated to the reported enjoyment. A fuzzy model inspired by psychophysiology theory is introduced by Mandryk and Atkins [18]. They reported that high values of HR and GSR together with a smile detection from an electromyography (EMG) in jaw are correlated to high values of arousal and positive valence.

Rani et al. [19] used psychophysiological measures such as HR and GSR to discriminate anxiety level and adjusted a Pong game to respond accordingly. In the approach proposed, they handle the problem of enjoyment maximization by appropriately minimizing the anxiety level.

Tijss et al in [20] showed values of skin conductance, HR and respiration to be statistically correlated to different difficulty levels of PacMan. The correlation with enjoyment state of a player is not directly calculated but it is inferred from a questionnaire analysis. Prediction models based on physiological state (HR, GSR) have also been proposed for potential entertainment augmentation in computer games [21].

In this work, we have applied some of the techniques presented in literature for physiological signal analysis for video game enjoyment evaluation. Preference learning techniques have been applied as presented by Yannakakis [9].

## II. EXPERIMENTAL SETTING

We propose a new gaming experimental protocol, tested on a car-racing computer game, in which affective computing techniques can be applied and validated. The protocol was designed to produce an affective computing benchmark dataset, that could be used also for further developments. This dataset is composed by physiological data, questionnaire answers regarding user data (i.e., game experiences and race preferences), game logs and 2 video camera recordings. 75 volunteers (60 males and 15 females) aged from 18 to 30

years old (57 from 19-25, 18 from 26-30) took part in this study.

### A. Task Design

The cognitive task in the experiment concerns playing a video game. This makes it possible to reach a high repeatability and a high level of involvement among participants. TORCS [2] was chosen as reference game for the following reasons: it is a video game that requires the player to be sitting in front of a computer, therefore subjects experiment emotionally different situations characterized by a similar physical activity and the effects of movement artifacts on acquired data are negligible differently from what happened in [9] where the subjects had to move and jump; this game is an open source project, therefore, it has been possible to implement custom logging and AI for opponent drivers; it is easy enough, even for an inexperienced player, thus the game experience can be kept as homogeneous as possible among subjects involved in the experiment.

During a game session, each participant played 7 races versus one computer driver that is the only opponent during the race. The opponent skill has been changed among races considering that this has a high potential impact on player emotional state, and that it can be easily adapted in a real-time affective loop to maintain the enjoyment level on the player according to the general principles of game engagement proposed by Malone [11].

Three classes of game scenarios have been considered and a customized opponent driver has been implemented to match the skill of the player. It modulates its speed to keep a given distance from the human driven car. We call W (Winner) the driver that is more skilled than the player and that has the goal to keep a distance of +100 m (relative distance between the cars within the current track) from the player. C (Challenging) is the driver that is as skilled as the player and tries to keep a distance of 0 m from the player. Finally L (Loser) is the driver that is less skilled than the player and that keeps a distance of -100 m from the player. According to a priori considerations, the second variant of the race could be considered really challenging, and more interesting for the player. Race parameters such as type of track, environmental details, car model, and number of opponents have been chosen to keep the game easy to play and to make the opponent skill being the main difference among races.

### B. Experimental Protocol

Most of the choices in the experimental protocol have been made to maximize the focus of the player on the task. The environment where the experiment took place has been conceived with the purpose of isolating the player and maximizing the game immersion so that no external event could influence the subject physiological state. The setting was a small room with a computer placed on a desk. The player was sitting in front of the monitor and was interacting with the computer through standard mouse and keyboard.

No other people were in the same room and the operator monitored the experiment from an external site.

Ahead of the experiment, all participants have been asked to fill out a general questionnaire, presented in computer-based form, and used to gather information about their experience with video games, game preference, TORCS prior knowledge, and personal data such as age and handedness.

Then, Participants have been fitted with sensors to measure peripheral physiological activity as explained in Section II-C. The players were asked to wear a headphone to guarantee a deeper game involvement through race sounds. After this setup phase, players were left alone listening to a relaxing music (i.e., sounds from nature) with the purpose of both decreasing the stress and the initial excitement for the test and bringing all the subjects to a similar starting condition.

Cameras and physiological signal acquisition were started while the players were waiting. Any misplacing of sensors was checked by the operator from the external site by looking at the real time signals. The subjects have been instructed to minimize movements during the task to avoid artifacts. To increase subject involvement during the game, players have been told that they were competing for a prize. Prizes were given basing on a series of parameters including in-game performance, but also on physiological features, so that potential advantages of skilled player were reduced. Note that, from this moment on, to avoid the effect of covert communication [22], no further interaction between operator and subject occurred. The protocol was carried on by an automatic script on the computer that started each race and managed the questionnaire (see Section II-D).

After about one minute of relaxing music, the participants were asked to read the instructions and then, to start the trial by pressing a button. At the end of each race, starting from the second one, the participants were asked by a script to express, via a computer-based form, the preference between the race just played and the previous one. To minimize any potential order effect on physiological and self-reported data, each pair of game variants have been presented in both orders. The sequence of driver classes was as follows: W C L W L C W. With this sequence, all permutations pairs of classes W, C, L could be voted by the player once. The duration of each race was 3 minutes. This provided enough time to eliminate past race effects on physiological signals and to produce a new arousal level before the overcoming of boredom caused by excessive race length.

The total time of a session was about 30 minutes, i.e., 21 minutes (7 races  $\times$  3 min.) of racing and about 7 minutes of setup, question answering and resting.

### C. Acquired data

In this protocol four types of data have been acquired: physiological data, questionnaire answers (presented in Section II-D), game logs and video camera recordings.

Physiological data were gathered using the ProComp Infiniti device [23]. This device captured 5 physiological signals: BVP, Electrocardiogram (ECG), GSR, Respiration (RESP) and Temperature (TEMP). A sample rate of 256Hz

TABLE I  
EXTRACTED FEATURES

| Feature of $x$ | Description                               |
|----------------|---|
| $x_m$          | Mean                                      |
| $x_v$          | Variance                                  |
| $x_{min}$      | Min value                                 |
| $x_{max}$      | Max value                                 |
| $x_D$          | Max - Min                                 |
| $x_{tm}$       | Time of Min value                         |
| $x_{tM}$       | Time of Max value                         |
| $x_{dT}$       | $\Delta T$ of Max and Min                 |
| $x_{fd}$       | Mean Absolute value of first differences  |
| $x_{sd}$       | Mean Absolute value of second differences |
| $x_{ct}$       | Trend                                     |
| $x_{acf}$      | AutoCorrelation function at 10s           |

has been used except for ECG and BVP signals that were sampled at 2048Hz. The hand not used for interacting with the game was fitted with GSR, BVP and TEMP sensors. The 3 terminal ECG sensor were placed around the chest, as well as the RESP sensor.

Based on previous literature [24], [25], [26], [9], several derived signals have been extracted from the basic ones at the original sampling rate. Heart rate has been derived both from ECG ( $HR_{ecg}$ ) and BVP ( $HR_{bvp}$ ); magnitude (SM) and duration (SD) of signal variation has been derived from GSR; inspiration/expiration time ( $inTime$ ,  $outTime$ ), apnea in/out time ( $apneau$ ,  $apnealow$ ) and respiration interval ( $rTime$ ) have been extracted from respiration signal; upper/lower envelope of BVP ( $BVP_{up}$ ,  $BVP_l$ ) and their difference ( $BVP_d = BVP_{up} - BVP_l$ ) have been also computed.

A feature vector  $F = [f_1 f_2 \dots f_D] \in \mathcal{R}^D$  has been finally obtained by the union of features described in Table I computed for each mentioned signal during each race. We assume that the first part of each race is subject to transitory phenomena due to the transition from a race to the next one. Thus, these features have been computed by considering only the last 60 seconds of each race.

A log file containing timestamp and some game status variables was saved during each race. Note that information regarding the TORCS state has not been used to obtain the results reported in this paper, but the timestamps have been used for synchronization between races and physiological data. Two video cameras recorded the environment in which the player acted too. A frontal camera captured the player's face, the second camera was placed at the top right back corner of the room, with respect to player, and captured the player actions and the game output from the monitor. All captured frames have been associated with a timestamp that is used for synchronization with the other signals. These data have not been considered in the analysis presented in this paper, but will be used by further research activities.

### D. Questionnaire

Enjoyment preference between races have been collected. At the end of each race, the subject was asked whether he/she enjoyed more the last race or the previous one. A

pairwise preference scheme (2-alternative forced choice: 2-AFC) has been used in self reports. 2-AFC offers a main advantage to acquire objective enjoyment: it normalizes the different conception of enjoyment among subjects and it allows a fair comparison between the answers of different subjects. Since we are concerned with finding a general model for the relationship between physiological features and reported entertainment preferences that generalizes over different players, 2-AFC is preferred with respect to other approaches, such as ranking [27].

### III. METHODS

Because of the lack of absolute ranked answers about player subjective enjoyment, canonical classification methods based on learning a target output is inapplicable since the target output is not defined.

Several techniques that learn from a set of pairwise preferences exist. Such algorithms are based on Gaussian processes [28], support vector machines (SVNs) [29], and evolving artificial neural networks (ANNs) [9]. In this work we focused on a linear approach developing a linear classifier, based on subject reported preferences and physiological features, which has similar performance, simpler structure, and lower computational demand.

#### A. Preliminary Statistical Analysis

A statistical analysis has been performed to understand the relationship between physiological features and reported enjoyment. In the first part of this analysis a Pearson's chi-square test [30] has been performed to establish whether the null hypothesis of independence between a feature and the reported preference could be accepted or rejected. When the null hypothesis is rejected, the feature and the preference can be considered not independent, but the strength of the true relationship is still unknown.

Then, a correlation analysis has been performed as proposed in [9]. We use Cohen's Kappa coefficient [31] to evaluate correlation between features and reported enjoyment defined as follows:

$$k = \frac{Pr(a) - Pr(e)}{1 - Pr(e)} \quad (1)$$

where  $Pr(a)$  is the relative observed agreement between the user preference and the difference of the mean of a single physiological feature between a pair of races;  $Pr(e)$  is the hypothetical probability of chance agreement, using the observed data to calculate the probabilities of each feature to be randomly correlated. If the preferences and the variation of mean values between pair of races are in complete agreement then  $k = 1$  (or  $k = -1$  in case of anticorrelated features). If there is no correlation (other than what would be expected by chance) then  $|k| \leq 0$ .

#### B. Preference Learning

Preference learning [6] is a technique that aims to predict the subject's preference among different elements. A subject expresses a preference  $A \succ B$  (we say  $A$  is preferred

over  $B$ ) between two elements  $A$  and  $B$  when he/she is able to order them with respect to a personal preference criterion. Each element is characterized by a set of features  $F = [f_1 f_2 \dots f_D] \in \mathcal{R}^D$ . The goal of preference learning is to estimate a preference function  $P$  that respects the set of  $N$  constraints:

$$\text{if } A_i \succ B_i \quad i = 1 \dots N \quad \text{then } P(F_i^A) > P(F_i^B)$$

Where  $F_i^A$  and  $F_i^B$  are the features vectors of elements  $A$  and  $B$  respectively in the  $i$ -th comparison and  $N$  is the total number of comparisons. Preference learning is a general approach and can be used to model subject's preference from physiological data. There are different techniques that can be used to estimate  $P$  depending on the function used. The Large Margin Algorithm (LMA) [29] can be used as linear preference learning classifier, and comes from Support Vector Machine theory (SVMs). This algorithm considers a linear combination of individual features  $F$  as emotional preference function  $P(F) = FW^T$  where the weight vector  $W = [w_1 w_2 \dots w_D]$  binds the user preferences to the physiological features. This method was first applied by Fiechter and Rogers [29] to a routing problem where the particular structure of the problem lead to a simplification of SVM approach to a linear problem. The main simplification was given by the fact that, in routing problems, preference decreases with costs represented by features  $F$ . Thanks to this hypothesis it was possible to add a new set of constraints  $w_j \geq 0 \quad j = 1, \dots, D$  that brought the quadratic problem into a linear optimization problem.

This method has been applied also to model subject preferences from physiological features as reported in [9]. However, features in our problem cannot be considered as costs and thus, the hypothesis  $w_j \geq 0 \quad j = 1, \dots, D$  would lead to a non optimal solution: weights greater than zero are given only to positively correlated physiological features i.e., negatively correlated features are ignored. We propose then to use a linear approach for preference learning based on Linear Discriminant Analysis (LDA) A.K.A Fisher's projection [8].

Given a set of  $N$  race pairs  $R_i^A \succ R_i^B \quad i = 1, \dots, N$  where the subject prefers  $R_i^A$  over  $R_i^B$  the goal is to estimate  $W$  in such a way that the user preference is preserved:

$$\text{if } R_i^A \succ R_i^B \quad i = 1, \dots, N \quad \text{then } F_i^A W^T > F_i^B W^T$$

where  $F_i^A$  and  $F_i^B$  are the feature vectors associated to  $R_i^A$  and  $R_i^B$  respectively. We can rewrite the previous inequality as  $(F_i^A - F_i^B)W^T = F_i^d W^T > 0$ . Where  $F_i^d$  is the feature difference between preferred and not preferred races of pair  $i$ . We thus reformulate the problem of estimating  $W$  as a linear classification problem by considering the data set  $X = \{x_i | i = 1 \dots N\}$ ,  $C = \{c_i | i = 1 \dots N\}$ , where  $x_i = [F_i^d, -F_i^d]$  and  $c_i = 0, 1$  (i.e., we assign class 0 to positive examples  $F_i^d$  and class 1 to negative examples  $-F_i^d$ ). The problem can be solved with Fisher's projection, which finds a projection direction  $W$  in which classes are well separated. In this way,  $W$  can be used to predict one of the two classes by evaluating the inequality  $XW^T < K$  (in our case  $K=0$  since the mean of our data is 0).

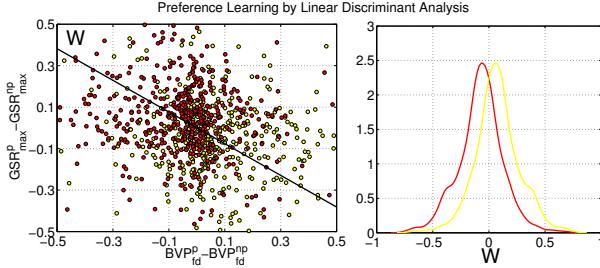


Fig. 1. Preference Learning by Linear Discriminant Analysis. A 2-D scatter plot of data from  $BVP$  and  $GSR$  is presented on the left. Sample points belonging to different classes have different colors. The black line  $W$  represents the direction computed by LDA in which data are projected. Probability densities of data by class with respect to the projection direction  $W$  is presented on the right. The direction  $W$  found by LDA is the one that best separates such densities.

To obtain Fisher's projection, we first define a within-class scatter matrix  $S_c$  as:

$$S_c = \sum_{c \in [0,1]} \sum_{x \in c} (x - \mu_c)(x - \mu_c)^T \quad (2)$$

where  $\mu_c$  is the mean value of sample points that belong to class  $c$ . Then we define a between-class scatter matrix

$$S_b = \sum_{c \in [0,1]} N_c (\mu - \mu_c)(\mu - \mu_c)^T \quad (3)$$

where  $N_c$  is the number of sample points in the class  $c$  and  $\mu$  is the mean value of all sample points. The best  $W$  that separates all the classes is the one that maximizes

$$J(W) = \frac{W^T S_c W}{W^T S_b W}. \quad (4)$$

Finding  $W$  is a one step process that requires much less time than running the optimization algorithm required by LMA.

Fisher's projection produces good classification performance when the distributions of data belonging to the same class are unimodal and classes are well separated. Figure 1 shows an example of application of LDA (based on real data obtained during our experiment) to classify the preferences of a subject by using two biological features  $BVP_{fd}$  and  $GSR_{max}$ . We can observe that both the distributions of  $BVP_{fd}^p - BVP_{fd}^{np}$  and  $GSR_{max}^p - GSR_{max}^{np}$  are unimodal but they are also partially overlapping, thus some of the preferences will be misclassified. Since all the data we are using in this work have an unimodal distribution similar to the one shown in Figure 1, the obtained performance depends on how well classes are separated.

#### IV. DATA ANALYSIS

In this section, the performance of the model for predicting reported preference is discussed. First we introduce the statistical analysis, then we present the results of the linear classifier based on LDA technique. The result will be finally extended by using different features selection methods.

#### A. Statistical Analysis

Results from the statistical analysis indicate that not all the features are dependent on the preference. This is confirmed by the fact that all p-values obtained from Pearson's  $\chi^2$  test are close to 0 ( $< 10^{-5}$ ); thus, we cannot accept the hypothesis of independence. The second part of the analysis characterizes the kind of dependence between features and preference by using the correlation coefficients  $k$ . The results of this analysis are presented in Table II where correlation coefficients  $k$  and  $\chi^2$  values from Pearson's  $\chi^2$  test are shown. The table is organized as follows: features derived from the same physiological signal are grouped together. For each group, the 5 most correlated features (higher absolute values of  $k$ ) are shown.

The physiological measurement that best correlates with reported user enjoyment is  $GSR$ , which achieved a correlation coefficient  $k = 0.332$  followed by  $BVP$  with  $k = -0.285$ . This result indicates that players tend to prefer games in which features from  $GSR$  increase and features from  $BVP$  decrease. Similar findings have been reported by Mandryk et al in [32] where  $GSR$  values resulted correlated with fun. Features related to respiration reported in Table II, represent time differences between respiration events and have also high negative correlation with enjoyment (i.e.,  $apnealow_m$  has a  $k = -0.234$ ). Thus, players preferred games in which breathing rate is increased (the time is decreased). Finally, good values of correlation have been obtained for  $HR$  ( $k = 0.166$ ) and temperature ( $k = -0.197$ ). Similar correlation results are reported also by Yannakakis and Hallam in [9] in their experiment on a physical playground. Our experiments have been performed on a computer video game that does not require physical activity, thus, features that best match user preferences in our work do not completely match the ones that were previously reported in [9].

#### B. Classification by Linear Discriminant Analysis

In the previous section, we have shown significant correlation between physiological features and the reported subject enjoyment. In this section, we present a quantitative evaluation of how each physiological feature can be used independently to predict the user preference. For each feature  $f_j$  ( $j = 1 \dots D$ ), an emotional preference function  $P(f_j) = f_j w_j$  is estimated through LDA technique [8] as explained in the previous section. Note that, since we are using only one feature, the estimated  $w_j$  could be only  $-1$  or  $1$  depending on the type of correlation. Given a pair of races  $R^A, R^B$ , the function  $P(f_j)$  classifies  $R^A$  as more entertaining if  $P(f_j^A) > P(f_j^B)$  where  $f_j^A$  and  $f_j^B$  are the  $j$ -th feature of preferred and non preferred race respectively. The performance of the preference function is defined as the number of correct pairwise classifications with respect to the total number of pairs (i.e., Correct Classification Rate CCR).

To guarantee significant values of performance, a leave-one-subject out cross validation has been applied to the classification process as follows: data relative to one subject

TABLE II

CORRELATION OF FEATURE AND PREFERENCE. HIGHER ABSOLUTE VALUES OF K MEAN HIGHER CORRELATION BETWEEN THE SINGLE FEATURE AND THE USER PREFERENCE.

|             | Bvp    |          |            | HR     |          |             | GSR   |          |                  | Resp   |          |             | Temp   |          |  |
|-------------|--------|----------|------------|--------|----------|-------------|-------|----------|------------------|--------|----------|-------------|--------|----------|--|
| f           | k      | $\chi^2$ | f          | k      | $\chi^2$ | f           | k     | $\chi^2$ | f                | k      | $\chi^2$ | f           | k      | $\chi^2$ |  |
| $bvp_{fd}$  | -0.285 | 36.9     | $hr_{dT}$  | 0.166  | 12.5     | $gsr_{sd}$  | 0.332 | 49.7     | $apnealow_m$     | -0.234 | 24.6     | $temp_{ct}$ | -0.197 | 17.5     |  |
| $bvp_{sd}$  | -0.285 | 36.9     | $hr_m$     | 0.145  | 9.6      | $gsr_{fd}$  | 0.328 | 48.3     | $apnealow_{min}$ | -0.234 | 24.6     | $temp_{tm}$ | 0.188  | 15.9     |  |
| $bvp_v$     | -0.244 | 26.9     | $hr_{min}$ | 0.145  | 9.6      | $SM_{fd}$   | 0.310 | 43.2     | $rrate_{max}$    | -0.229 | 23.7     | $temp_v$    | -0.159 | 11.6     |  |
| $bvp_m$     | 0.232  | 25.2     | $hr_{max}$ | 0.111  | 5.6      | $SM_{sd}$   | 0.310 | 43.2     | $inrate_m$       | -0.213 | 20.4     | $temp_D$    | -0.161 | 11.8     |  |
| $bvp_{min}$ | 0.232  | 25.2     | $hr_{tm}$  | -0.103 | 4.7      | $gsr_{max}$ | 0.265 | 31.8     | $inrate_{min}$   | -0.213 | 20.4     | $temp_{fd}$ | -0.143 | 9.4      |  |

TABLE III

CORRECT CLASSIFICATION RATE USING LDA ALGORITHM AS LEARNING TECHNIQUE FOR SINGLE FEATURES.

|             | Bvp   |     | HR         |       | GSR         |       | Resp             |       | Temp        |       |
|-------------|-------|-----|------------|-------|-------------|-------|------------------|-------|-------------|-------|
|             | f     | CCR | f          | CCR   | f           | CCR   | f                | CCR   | f           | CCR   |
| $bvp_{fd}$  | 0.638 |     | $hr_{dT}$  | 0.582 | $gsr_{sd}$  | 0.667 | $apnealow_m$     | 0.616 | $temp_{ct}$ | 0.596 |
| $bvp_{sd}$  | 0.638 |     | $hr_m$     | 0.571 | $gsr_{fd}$  | 0.664 | $apnealow_{min}$ | 0.616 | $temp_v$    | 0.582 |
| $bvp_v$     | 0.618 |     | $hr_{min}$ | 0.571 | $SM_{fd}$   | 0.656 | $rrate_{max}$    | 0.611 | $temp_D$    | 0.582 |
| $bvp_m$     | 0.613 |     | $hr_{max}$ | 0.556 | $SM_{sd}$   | 0.656 | $inrate_m$       | 0.604 | $temp_{fd}$ | 0.573 |
| $bvp_{min}$ | 0.613 |     | $hr_{tm}$  | 0.551 | $gsr_{max}$ | 0.636 | $inrate_{min}$   | 0.604 | $temp_{sd}$ | 0.569 |

have were and remaining data are used for training; the LDA is trained on the training data set and the performance of the model is tested on the data of the removed subject. These steps are iterated for each subject and the mean values are reported in Table III. The table is organized as follows: features derived from the same physiological signal are grouped together. For each group are shown the 5 features that give the best classification performance using LDA. Note that the values are consistent with the analysis of correlation reported in Table II: The best classification performance ( $CCR = 0.667$ ) is achieved by GSR, which is the most correlated signal having the highest absolute value of  $k$ . BVP obtained a  $CCR=0.638$ , respiration reported  $CCR=0.616$  and finally, for HR and temperature the  $CCR$  was 0.582 and 0.596, which are closer to 0.5 characteristic of a random classifier.

The results confirm that the Preference Learning approach can be successfully applied to model player preference in video games where the physical activity is kept as constant as possible over different type of games. Moreover, these results support the proposed experimental protocol as a successful way to produce reliable and replicable data for affective computing experiments.

### C. Improvements by Considering Multiple Features

To improve single feature performance, a linear combination of all physiological features  $F$  by LDA has been used to predict subject enjoyment as explained in previous section. This is a generalization of the analysis performed over a single feature since a combination of features may introduce information useful for classification. Leave-one-subject out cross validation has been used to evaluate the performance. The LDA algorithm takes the full set of features as input and tries to find the best combination of weights  $W$  that separates classes. Correct classification rate achieved using all features

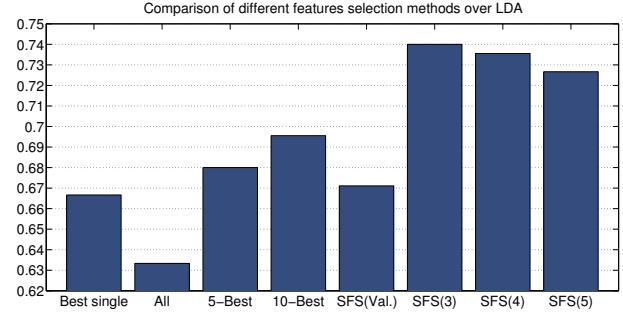


Fig. 2. Classification performance achieved with different subset of features: Best single feature, All features, Selection of best 5 or 10 features (5-Best and 10-Best), SFS on validation data (SFS(Val.)), SFS with pseudo-intersection ( $n=3,4,5$ )

is 0.633. This result is lower than the one obtained with the best single feature. This is likely, since some of the features might have introduced noise instead of adding useful information.

Since including all features did not provide better performance than the best single feature, we tried to find a subset of features that performed better than single feature classification. We selected the first 5 and 10 features that individually gave the best CCR and the performance achieved was 0.68 for 5-best and 0.696 for 10-best. CCR is higher than the one obtained with the complete set likely because some of the noisy features are not included by this rough selection technique. A comparison of performances is reported in Figure 2

A better, still greedy, method for feature selection is the Sequential Feature Selection (SFS) [33] that finds the minimal subset of features that maximize the classification performance. It is a bottom-up algorithm where, at each

step, an additional feature is added to the current feature set. The selected feature is added if its marginal value with respect to the classification function is positive. To evaluate the performance of a feature selection algorithm like SFS, we used an external cross-validation procedure. This avoids the overfitting of selection for the current dataset. A K-fold (K=15) cross-validation by subject has been performed. Data belonging to all the players in the original dataset  $D_s$  have been randomly split into K folds  $D_{s_i}$   $i = 1 \dots 15$  (containing data of 5 subjects each). Folds have been then assembled into 15 data set containing training data  $D_{s_i}^{tr} = D_s \setminus D_{s_i}$  and validation data  $D_{s_i}^{val} = D_{s_i}$ . The SFS has been executed K times independently over each different data set. For each iteration, the best feature selection with respect to the training data  $D_{s_i}^{tr}$  has been evaluated on validation data  $D_{s_i}^{ts}$  in order to estimate the selection performance on previously unseen data. Each selection found on training has been evaluated with the same leave-one-subject out cross validation method presented in Section IV-A. Each independent run selected the best subset of features that obtained different performance on validation data with mean value 0.671 of correct preference prediction over all the data set. Due to the different type of cross validation, this estimate of performance results the most general and least overfitted, hence the most significant. This result, labeled as SFS(Val.), is compared in Figure 2 with other feature selection methods.

The performance over validation gives an estimation of how well the classifier would perform with previously unseen data. However, with the presented SFS approach it is not possible to know which is the best subset of features that maximizes the CCR since each fold may yield to a different selection. To give an indicative measure of performance of the best general subset of features, a pseudo-intersection of the selections produced by each of the 15 independent runs has been performed. The pseudo-intersection merges the features that have been selected by at least  $n$  independent runs of SFS. We compared pseudo-intersection feature subsets with  $n = 3, 4, 5, 6, 7$ . We finally tested the obtained selections over the full data set with the previous described leave-one-subject out cross validation. In Figure 2 SFS(3)=0.74, SFS(4)=0.7356 and SFS(5)=0.7267 represent results obtained by the pseudo-intersections when features where selected by 3, 4 or 5 runs out of 15. The highest value is obtained with SFS(3) since it uses almost all the features that each fold have selected. This is the highest performance we achieved through leave-one-subject out cross validation by applying a linear function for preference modeling on these data.

In Table IV, we reported the features selected by some of the used methods. In the case of pseudo-intersection the performance decreases as much as we increase the number of folds from which features have been selected. This is due to the fact that pseudo-intersection exploits the information coming from each independent run. The less runs are considered, the more performance is overestimated, since we are using more data to obtain the selection. However, the more a feature has been selected from a fold the more

TABLE IV  
SELECTED FEATURES BY DIFFERENT METHODS

| Type   | Selection   | Performance |
|--------|---|-------------|
| SFS(3) | $BVP_{fd}$ $BVP_{sd}$ $GSR_{fd}$ $GSR_{sd}$ $SM_v$<br>$SD_{fd}$ $inrate_m$ $outrate_v$ $apnealow_v$ | 74%         |
| SFS(4) | $BVP_{fd}$ $GSR_{fd}$ $SM_v$ $SD_{fd}$ $outTime_v$  | 73.56%      |
| SFS(5) | $BVP_{fd}$ $GSR_{fd}$ $SD_{fd}$ $outTime_v$   | 72.67%      |
| SFS(6) | $BVP_{fd}$ $GSR_{fd}$ $SD_{fd}$ $outTime_v$   | 72.67%      |
| SFS(7) | $BVP_{fd}$ $GSR_{fd}$ $outTime_v$   | 69.33%      |
| SFS(8) | $GSR_{fd}$  | 66.67%      |
| 5-Best | $BVP_{fd}$ $GSR_{sd}$ $GSR_{fd}$ $SM_{sd}$ $SM_{fd}$  | 68%         |

it is an invariant measure of preference among subjects. This is the case of  $BVP_{fd}$ ,  $GSR_{fd}$  and  $outTime_v$  that have been selected at least by 7 out of 15 runs of SFS, so they are likely to be invariant features among subjects to predict enjoyment. Table IV shows also how the selections produced by SFS are different with respect to the 5-Best or 10-Best approach. SFS obtains higher performance, fixed the number of used features (i.e., SFS(4) vs 5-Best), since it removes variables that are dependent on the ones that have been already selected (i.e.,  $GSR_{sd}$  and  $GSR_{fd}$  or  $SM_{sd}$  and  $SM_{fd}$ ) and introduces other variables that can be more useful even if they do not produce high performance when used alone.

By means of this analysis we have now a clear picture of which are the most relevant features that combined linearly can predict the reported preference of video game players with a CCR up to 0.74% on previously ~~unseen~~ data.

## V. CONCLUSION

We have presented a way to identify the preference of players among different variants of a video game. We have proposed a new video game scenario in which 3 different game conditions have been obtained by controlling the opponent skill. The proposed scenario requires the player to be sitting in front of a computer, therefore the effects of movement artifacts on acquired data are reduced. Moreover, thanks to the adaptive controller, the game experience during the test has been homogeneous among different subjects. A data set composed by physiological data, questionnaire answers regarding user data, game experiences, race preferences, game logs, 2 video camera recordings have been acquired with the purpose of building a benchmark data set in which different Affective Computing techniques can be applied and validated.

We performed correlation analysis between physiological data and subject preference, showing that features from  $GSR$  have a high correlation with player reported preferences ( $k = 0.332$ ,  $\chi^2 = 49.7$ ,  $p-value \approx 10^{-5}$ ). Similar results have been shown by Mandryk et al. in [32] and are slightly different from the ones presented by Yannakakis and Hallam in [9] probably due to the different nature of the task involved (computer video game vs physical playground). We have estimated a linear model of preference that maps a subset of physiological features to the preference level, through a novel approach that makes use of Linear Discriminant Analysis (LDA). Results showed that this model is able to

predict reported preference with an accuracy up to 0.74% on previously unseen data.

The analysis of questionnaires highlighted that for less than 42% of subjects there was a consistent agreement on the order of preference of game variants. That means that for them, a situation where the opponent is as skilled as the player was always preferred. However, the remaining players either did not answered coherently among different repetition of the same stimulus or they expressed a different order of personal preference. This interesting result indicates that the game preference is personal and it is hard to design a priori game experience (e.g., by changing opponent skills) that results suitable for everyone. This motivates the current research that aims to evaluate the enjoyment from biological signals without any assumption on players game preferences.

Player satisfaction from physiological data, is perhaps one of the most promising application area of affective computing. Classic and canonical games could be enhanced to adapt to the player affective states, and entirely new types of games could be created. Results from this experiments can be used as starting point for a follow up experiment in which game experience is modified in realtime with the purpose of keeping the player satisfaction to a high level.

#### ACKNOWLEDGMENTS

The research activity described in this paper has been partially supported by IIT - Italian Institute of Technology.

#### REFERENCES

- [1] P. Ekman, R. Levenson, and W. Friesen, "Autonomic nervous system activity distinguishes among emotions," *Science*, vol. 221, no. 4616, pp. 1208–1210, 1983.
- [2] "The open racing car simultaor website," <http://torcs.sourceforge.net/>.
- [3] G. Yannakakis and J. Hallam, "Towards capturing and enhancing entertainment in computer games," *Lecture Notes in Computer Science*, vol. 3955, p. 432, 2006.
- [4] G. Yannakakis, H. Lund, and J. Hallam, "Modeling children's entertainment in the playware playground," in *Proceedings of the IEEE Symposium on Computational Intelligence and Games*, 2006, pp. 134–141.
- [5] J. Furnkranz and E. Hullermeier, "Pairwise preference learning and ranking," *Lecture Notes in Computer Science*, vol. 2837, pp. 145–156, 2003.
- [6] J. Doyle, "Prospects for preferences," *Computational Intelligence*, vol. 20, no. 2, pp. 111–136, 2004.
- [7] G. Yannakakis, "Preference Learning for Affective Modeling," in *Proceeding of the international conference on Affective Computing and Intelligent Interaction, ACII 2009*, Amsterdam, Netherland, 2009.
- [8] R. Duda, P. Hart, et al., *Pattern classification and scene analysis*. Wiley New York, 1973.
- [9] G. Yannakakis and J. Hallam, "Entertainment modeling through physiology in physical play," *International Journal of Human-Computer Studies*, vol. 66, no. 10, pp. 741–755, 2008.
- [10] ——, "Capturing Player Enjoyment in Computer Games," *Computational Intelligence (SCI)*, vol. 71, pp. 175–201, 2007.
- [11] T. Malone, "What makes computer games fun?" in *Proceedings of the joint conference on Easier and more productive use of computer systems.(Part-II): Human interface and the user interface-Volume 1981.* ACM New York, NY, USA, 1981.
- [12] M. Csikszentmihalyi, *Flow: The psychology of optimal experience*. Harper & Row New York, 1990.
- [13] J. Cacioppo, L. Tassinary, and G. Berntson, *Handbook of Psychophysiology*. New York, NY: Cambridge University Press, 2000.
- [14] R. Picard, E. Vyzas, and J. Healey, "Toward machine emotional intelligence: analysis of affectivephysiological state," *IEEE transactions on pattern analysis and machine intelligence*, vol. 23, no. 10, pp. 1175–1191, 2001.
- [15] C. Lisetti, F. Nasoz, C. LeRouge, O. Ozyer, and K. Alvarez, "Developing multimodal intelligent affective interfaces for tele-home health care," *International Journal of Human-Computer Studies*, vol. 59, no. 1-2, pp. 245–255, 2003.
- [16] J. Kim and E. Andrè, "Emotion recognition based on physiological changes in listening music," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 30 (12), pp. 2067–2083, December, vol. 30, no. 12, pp. 2067–2083, December 2008.
- [17] R. Mandryk, K. Inkpen, and T. Valvert, "Using psychophysiological techniques to measure user experience with entertainment technologies," *Behaviour & information technology(Print)*, vol. 25, no. 2, pp. 141–158, 2006.
- [18] R. Mandryk and M. Atkins, "A fuzzy physiological approach for continuously modeling emotion during interaction with play technologies," *International Journal of Human-Computer Studies*, vol. 65, no. 4, pp. 329–347, 2007.
- [19] P. Rani, N. Sarkar, and C. Liu, "Maintaining optimal challenge in computer games through real-time physiological feedback," in *Proceedings of the 11th International Conference on Human Computer Interaction*, 2005, pp. 184–192.
- [20] T. Tijs, D. Brokken, and W. IJsselsteijn, "Dynamic game balancing by recognizing affect," in *Proceedings of the 2nd International Conference on Fun and Games*. Springer, Berlin, 2008, p. 93.
- [21] S. McQuiggan, S. Lee, and J. Lester, "Predicting user physiological response for interactive environments: an inductive approach," in *Proceedings of the 2nd Artificial Intelligence for Interactive Digital Entertainment Conference*, 2006, pp. 60–65.
- [22] R. Rosenthal, "Covert communication in laboratories, classrooms, and the truly real world," *Current Directions in Psychological Science*, pp. 151–154, 2003.
- [23] "Thought technology ltd., 2002. website," <http://www.thoughttechnology.com/>.
- [24] A. Bonarini, L. Mainardi, M. Matteucci, S. Tognetti, and R. Colombo, "Stress recognition in a robotic rehabilitation task," in *Proc. of "Robotic Helpers: User Interaction, Interfaces and Companions in Assistive and Therapy Robotics"*, a Workshop at ACM/IEEE HRI 2008, vol. 1. Amsterdam, the Netherlands: University of Hertfordshire, March 2008, pp. 41–48.
- [25] S. Tognetti, C. Alessandro, A. Bonarini, and M. Matteucci, "Fundamental issues on the recognition of autonomic patterns produced by visual stimuli," in *Proceeding of the international conference on Affective Computing and Intelligent Interaction, ACII 2009*, Amsterdam, Netherland, 2009.
- [26] R. Picard, E. Vyzas, and J. Healey, "Toward machine emotional intelligence: Analysis of affective physiological state," *IEEE transactions on pattern analysis and machine intelligence*, pp. 1175–1191, 2001.
- [27] R. Likert, "A technique for the measurement of attitudes," *Archives of Psychology*, vol. 140, pp. 1–55, 1932.
- [28] W. Chu and Z. Ghahramani, "Preference learning with Gaussian processes," in *Proceedings of the 22nd international conference on Machine learning*. ACM, 2005, p. 144.
- [29] C. Fiechter and S. Rogers, "Learning subjective functions with large margins," in *ICML 2000: Proceedings of the Seventeenth International Conference on Machine Learning*, 2000, pp. 287–294.
- [30] H. Chernoff and E. Lehmann, "The use of maximum likelihood estimates in  $\chi^2$  tests for goodness of fit," *The Annals of Mathematical Statistics*, pp. 579–586, 1954.
- [31] J. Cohen, "Coefficient of agreement for nominal scales. Educational and Psychological Measurement," *Psychological bulletin*, vol. 20, pp. 37–46, 1960.
- [32] R. Mandryk, M. Atkins, and K. Inkpen, "A continuous and objective evaluation of emotional experience with interactive play environments," in *Proceedings of the SIGCHI conference on Human Factors in computing systems*. ACM, 2006, pp. 1027–1036.
- [33] P. Pudil, J. Novoviová, and J. Kittler, "Floating search methods in feature selection," *Pattern recognition letters*, vol. 15, no. 11, pp. 1119–1125, 1994.

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/225407995>

# Ranking vs. Preference: A Comparative Study of Self-reporting

Conference Paper · October 2011

DOI: 10.1007/978-3-642-24600-5\_47 · Source: dx.doi.org

---

CITATIONS

65

READS

109

2 authors, including:



Georgios Yannakakis

University of Malta

220 PUBLICATIONS 5,360 CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:



AutoGameDesign -- Sonancia Project [View project](#)



COMnPLAY-Science [View project](#)

# Rating vs. Preference: A comparative study of self-reporting

Georgios N. Yannakakis<sup>1</sup> and John Hallam<sup>2</sup>

<sup>1</sup> Center for Computer Games Research, IT University of Copenhagen, Rued Langgaards Vej 7, Copenhagen S, Denmark [yannakakis@itu.dk](mailto:yannakakis@itu.dk)

<sup>2</sup> Maersk Mc-Kinney Moller Institute, University of Southern Denmark, Campusvej 55, Odense, Denmark [john@mmt.sdu.dk](mailto:john@mmt.sdu.dk)

**Abstract.** This paper introduces a comparative analysis between rating and pairwise self-reporting via questionnaires in user survey experiments. Two dissimilar game user survey experiments are employed in which the two questionnaire schemes are tested and compared for reliable affect annotation. The statistical analysis followed to test our hypotheses shows that even though the two self-reporting schemes are consistent there are significant *order of reporting* effects when subjects report via a rating questionnaire. The paper concludes with a discussion of the appropriateness of each self-reporting scheme under conditions drawn from the experimental results obtained.

## 1 Introduction

Self-reporting provides the most direct approach to user experience annotation and affect detection. Quantitative reports via questionnaires offer unique properties for constructing computational models of reported user states (affective or cognitive) and ease the analysis of subjective assessment in user studies. Even though beneficial for cognitive and affective capture and modeling, such reporting has several limitations such as self-deception, intrusiveness and subjectiveness. The appropriateness of the reporting scheme used for affect detection is therefore vital for the validity of the obtained analysis.

This paper examines the relationship between two popular self-reporting schemes in user studies: self-reporting via *rating* (or scaling) and via *pairwise preference*. The two schemes are compared in two dissimilar game survey studies in which experiment participants are asked to post-report a set of affective states. For the comparison to be possible, pairwise preferences are inferred from the rating values and compared to the direct pairwise preferences. The two hypotheses the two questionnaire schemes are tested against are:

- H1: There is an inconsistency between reported preferences and reported rating.  
Rating responses do not match reported preferences.
- H2: The order of post-experience reporting has an effect on both rating and preference report schemes. Randomness exists in both self-report schemes.

The statistical analysis followed to test the above hypotheses suggests that while rating and preferences are consistent (with variant degrees of consistency), pairwise preferences are more appropriate detectors of user states, eliminating the subjective notion of scaling and effects related to reporting order.

## 2 Self-reporting

This paper focuses on **forced self-reports** obtained via questionnaires. Such a self-report scheme constrains the participant to specific questionnaire items which could vary from simple tick boxes to multiple choice items while both the questions and the answers provided could vary from single words to sentences. Two types of forced self-reports that are described in more detail below, define the framework of investigations in this paper: self-reports via **rating** (or scaling) and self-reports via **preferences**.

### 2.1 Rating

The vast majority of psychometric and user studies have adopted a type of rating report to capture the subjective assessment of the experiment participants ([11] among others). The most popular approach to rating reports is a form of a **Likert scale** [5] in which users are asked to rate an experience, an emotion or an interactive session. In most such studies Likert ratings are **usually averaged across users** before they are further analyzed. Such a practice has an impact on the ratings **losing their subjective nature** but also implies a knowledge of the scale that is beyond a relative rank order of data [11].

Among the limitations of rating ordinal scales, Linn and Gronlund [6] indicate the existence of personal bias which may (among others) occur when the subject is consistently using only part of the scoring scale, logical errors due to the confusion of the distinct items of an ordinal scale and the ability to use numerical information within scales which is affected by the subject's internal cognitive processes, cultural background, temperament, and interests [13]. There is also a large body of work suggesting the presence of *primacy* and *recency* order effects in Likert questionnaires (see [2] among others).

The authors are not aware of a reliable statistical test that validates the reliability of a rating questionnaire as a whole. Cronbach's alpha [4] (*inter alia*) is an estimate of internal consistency (or reliability) of sections of the questionnaire; Cohen's kappa [3] assesses rater agreement in nominal scales.

### 2.2 Preference

Reporting via pairwise preferences has recently attracted the interest of researchers in affective and cognitive modeling ([16, 14, 12] among others) since it minimizes the assumptions made about subjects' notions of highly subjective constructs such as emotions and allows a fair comparison between the answers of different subjects. Moreover, artifacts such as the subjective notion of rating/scaling are eliminated and lead to the construction of **generalisable and accurate computational models of affect via user preference modeling** [14].

A preference questionnaire scheme may ask for the pairwise or multiple preference of participants or even ask them to provide a preferred order. In this paper we investigate pairwise preferences and are inspired by the seminal work of Scheffe [10] and Agresti [11] for the analysis of paired comparisons.

### 3 User survey case studies

This section presents the main phases of the experimental procedure followed to obtain self-reported emotional or cognitive states of experiment participants via both rating and preference schemes. The reader is referred to [16] for more details on the experimental protocol used. The section concludes with the presentation of the two case studies considered in this paper.

#### 3.1 System Instrumentation

The interactive systems we investigated are instrumented based on controllable parameters identified by the designer. The selection of the parameters is based on their potential impact on the user's affective and cognitive states examined and thereby to the post-experience self-reporting. For instance, a controllable parameter in a game system could be the speed of the game.

For each parameter under investigation, a number of states (e.g. 'Low', 'High') are selected. The product of the number of states for each of the parameters defines the number of different system variants that will be examined. Given the proposed experimental design [16] each survey participant interacts with system variants in pairs (variant A and variant B) — differing in the levels/states of one or more of the selected controllable parameters — for a selected time window. To test for potential order effects each subject interacts with the aforementioned system variants in both orders. Each time a system variant is completed the subject is asked to rate a particular experience using both a rating and a pairwise preference reporting scheme (described below).

#### 3.2 Self-reported Post-experience

For rating questionnaires the question is expressed as: "The session felt  $E$ ." where  $E$  is the emotional state (e.g. frustration) under investigation. Two rating scales have been used in the experiments reported: a 20 point 0-10 scale, and a 1-5 scale. The 0-10 scale uses principles of the funometer [9]; subjects have to rate the experience in a thermometer-designed Likert scale. On the other hand, the answers in the 1-5 scale rating scheme are inspired by the game experience questionnaire (GEQ) [8]; numbers have following glosses: 1: *not at all*; 2: *slightly*, 3: *moderately*, 4: *fairly* and 5: *extremely*.

For pairwise preference questionnaires subjects are asked to fill in a questionnaire each time a pair of game sessions (variants) is finished. According to this scheme, the subject is asked to report whether the first variant felt more  $E$  than the second variant. Specifically, for each completed pair of system variants  $A$  and  $B$ , subjects report their preference regarding an emotional state,  $E$ , by selecting among the following 4-alternative forced choices (4-AFC):  $A$  [ $B$ ] felt more  $E$  than  $B$  [ $A$ ] (cf. 2-alternative forced choice); both felt equally  $E$ ; neither of the two felt  $E$ .

One of the limitations of the experimental protocol proposed is **post-experience**. Users report emotional states *after* playing games, which might generate memory-dependencies in the reports. Effects such as order of play and game learnability might also be apparent and interconnected to memory. The experimental protocol, however, is designed to test for order of play effects which, in part, reveal memory (report consistency over different orders) and learnability effects, if any. Lack of significant order effect provides evidence that the experimental noise generated in this way is random. Statistical analysis of the effect of order on subjects' emotional judgement indicates the level of randomness in subjects' preferences. **Randomness** is apparent when the subject's expressed preferences are inconsistent for the pair ( $A, B$ ) independently of the questionnaire-scheme used.

### 3.3 The Playware Case Study

The first case study presented concerns game play sessions followed by self-reporting sessions of children playing physical interactive games [16, 17]. The game, called '**Bug-Smasher**', designed using the Playware playground (interactive tiles) platform [7], is used here as the test-bed interactive system for investigating the relationship between self-reporting schemes. (The reader is referred to [16] for more details of Bug-Smasher).

Seventy six children, aged 8 to 10 years old, participated in the survey experiment. Each subject played a set of 90 second Bug-Smasher variants, differing with respect to two control parameters: the speed of the game and the spatial diversity of game opponents. Children were not interviewed but were asked to fill in a questionnaire, minimizing interviewing effects. Each subject was asked to rate each game via a 10-scale *funometer* [9] (in increments of 0.5) and after a pair of games were finished, to report a fun preference for the two games she played using a 2-AFC question, "which one of the two games was more fun?" The options offered for choice were "first" and "second".

### 3.4 The Maze-ball Case Study

A screen-based computer game, named Maze-ball, is used for the second experiment reported in this paper. **Maze-ball** [18] is a three-dimensional predator/prey game. The goal of the player (ball) is to maximize her score by gathering as many tokens, scattered in the maze, as possible while avoiding being touched by a number of opponents in a predefined time window of 90 seconds. Further details about Maze-Ball and experimental design can be found in [18].

Thirty six subjects aged from 21 to 47 years participated to the experiment. Each subject played a predefined set of eight games for 90 seconds each; the games differ in the virtual camera profile embedded. For each completed game and pair of games  $A$  and  $B$ , subjects report their emotional preference using a 5-point Likert scale based on GEQ [8] followed by a 4-AFC pairwise preference protocol. The emotional states,  $E$ , examined comprise *fun*, *challenge*, *boredom*, *frustration*, *excitement*, *anxiety* and *relaxation*. The selection of these seven states is based on their relevance to computer game-playing and player experience.

### 3.5 Case Study Dissimilarities

The main dissimilarities between the two case studies are that in Playware 1) subjects are children (aged: 8 to 10), 2) a pen-and-paper (instead of a digital) questionnaire is used, 3) a rather broad ordinal scale from 1 to 10 is used for the rating scheme, 4) 2-AFC (instead of 4-AFC) is used for the preference scheme; 5) and subjects are asked only one question, about fun. Cognitive load during the reporting phase in the Playware experiment appears less due to the presence of only one question. Moreover, the broad rating scale used may allow for a better approximation of the level of reported fun.

Comparison of findings across the two case studies is not appropriate given the large number of dissimilarities in terms of gameplay interaction and experimental protocol. However, collectively, they provide two related but different studies of post-experience reporting in games and their analysis assists the understanding of the interplay between reported preferences and rating across different schemes.

## 4 Results and analysis

This section presents the results of the statistical analysis for testing our hypotheses in the two case studies. First, the statistics employed to test our research hypotheses H1 and H2 are outlined below.

### 4.1 H1 test statistic

To measure the degree of agreement between the rating and preference self-reports we calculate the correlation coefficients between them, obtained using  $c(\mathbf{z}) = \sum_{i=1}^N \{z_i/N\}$  following the statistical analysis procedure for pairwise preference data introduced in [15].  $N$  is the total number of incidents to correlate, and  $z_i = +1$ , if rating reports match preference reports and  $z_i = -1$ , if rating and preference reports are mismatched in the game pair  $i$ . In the calculation of  $c(\mathbf{z})$  we only take into account *clear* preferences and ratings of participants. That is, we only consider game pairs in which both a clear preference (i.e.  $A > B$  or  $A < B$ ; 2-AFC) and a clear rating (i.e.  $A > B$  or  $A < B$ ) are expressed. The p-values of  $c(\mathbf{z})$  are obtained via the binomial distribution.

### 4.2 H2 test statistics

To measure whether the order of play affects the player's judgement of rating or pairwise preference for affective states, we follow the order testing procedure described in [15], based on the number of times that the subject prefers the first (primacy effect) or the second (recency effect) game in both pairs. Briefly, the order test statistic is calculated as  $r_o = (K - J)/N$ , where the subject prefers (either via rating or preference) the first session in both pairs  $K$  times and, the second session in both pairs  $J$  times. The greater the absolute value of  $r_o$  the more the order of play tends to affect the subjects' judgement of interest.  $r_o$  is trinomially-distributed under the null hypothesis..

In addition to the  $r_o$  value we calculate the  $r_c = (K + J)/N$  test statistic, which yields a measure of reporting consistency with respect to order. The obtained  $r_c$  value

lies between 0 (reporting is consistent) and 1 (reporting is inconsistent) and is binomially-distributed with mean 0.5 under the null hypothesis.

The order effects are calculated solely on clear preferences (i.e. when  $A \succ B$  or  $A \prec B$ ) and ratings (i.e. when  $A > B$  or  $A < B$ ) in both pairs played in both orders. The significance level used in this paper is 5%.

### 4.3 Playware

The total number of game pairs with valid reported data is 105 in the Playware experiment. To calculate the statistics we exclude the 35 game pairs in which an equal rating is reported. The correlation between reported rating and preference  $c(z) = 0.857$  ( $p$ -value  $= 4.002 \cdot 10^{-10}$ ) indicates a statistically significant effect and rules out H1.

**Order Effect Analysis** Statistical analysis of the subjects' answers shows that no significant order effect occurs ( $r_o = -0.102$ ,  $p$ -value = 0.224) when preferences are reported, which rules out hypothesis H2. However, a significant effect of playing order on rating reports is found ( $r_o = -0.3809$ ,  $p$ -value = 0.0097) which indicates a tendency to consistently rate the second game higher. The insignificant order effect for reported *preferences*, in part, demonstrates that effects such as a subject's possible preference for the very first game played and the interplay between reported fun and familiarity with the game are statistically insignificant. On the other hand, the significant order effect for reported rating suggests that the order of play influences reporting when the rating scheme is used.

The  $r_c$  values for rating and preferences are 0.476 ( $p$ -value = 0.124) and 0.338 ( $p$ -value = 0.009), respectively, suggesting that only the preference reports appear to be ~~consistent with respect to order~~.

**Analysis & Conclusions** The first case study provides indications of **inconsistency between rating and preference reports**. While the two are statistically correlated ( $c(z) = 0.857$ ) there are several instances (16.6% of the data samples) in which preferences do not agree with their corresponding rating.

The inconsistency between the two report schemes may have occurred for a number of reasons including self-deception, cognitive load, question understanding in small children etc. A first analysis of the effect of order of game interaction shows that significant order effects exist only in reported ratings which in turn suggests existence of **randomness** when expressing rating choices for the game sessions attempted. Moreover, the consistency of reports with respect to order,  $r_c$ , appears to be significant for the preference reports only.

### 4.4 ~~Maze-Ball~~

For the Maze-Ball case study we follow the same statistical analysis presented above for the Playware game. The total number of valid game pairs examined in the Maze-Ball survey is 56 and the matching correlation ( $c(z)$ ) values between rating and preference reports for the Maze-Ball test-bed are depicted in Table 1.

**Table 1.** Maze-ball: Correlation coefficient values ( $c(z)$ ) between rating and clear preferences (2-AFC), and order of play ( $r_o$ ) and consistency ( $r_c$ ) correlation coefficients for all investigated emotional states  $E$ . Significant effects appear in bold.

| $E$         | $c(z)$       | $r_o$         |            | $r_c$        |              |
|-------------|--------------|---------------|------------|--------------|--------------|
|             |              | Rating        | Preference | Rating       | Preference   |
| Fun         | <b>0.925</b> | <b>-0.375</b> | -0.150     | 0.375        | 0.450        |
| Challenge   | <b>0.733</b> | <b>0.300</b>  | -0.222     | 0.500        | 0.444        |
| Frustration | <b>0.878</b> | -0.083        | -0.066     | <b>0.250</b> | <b>0.187</b> |
| Anxiety     | <b>0.619</b> | 0.200         | -0.222     | <b>0.200</b> | 0.444        |
| Boredom     | <b>0.666</b> | <b>-0.333</b> | -0.111     | 0.333        | <b>0.111</b> |
| Excitement  | <b>0.642</b> | -0.200        | -0.117     | <b>0.200</b> | <b>0.312</b> |
| Relaxation  | <b>0.652</b> | <b>-0.250</b> | 0.052      | 0.250        | 0.368        |
| Total       | <b>0.744</b> | -0.090        | -0.112     | <b>0.309</b> | <b>0.353</b> |

It appears there is a varying degree of consistency between rating and preference reports depending on the affective state (question asked). Overall 2-AFC preference reports appear to be consistent with rating reports. For the fun, frustration and challenge reports the two schemes are highly correlated (correlation higher than 0.7) whereas for the other four questionnaire items the correlation lies within the 0.6-0.7 interval; however, in all seven affective state questionnaire items, the correlation is statistically significant ruling out H1. These effects might be linked to the order of question items appearing in the questionnaire which is equivalent to the order the emotional states that appear in Table 1; the questions about excitement and relaxation, for instance, were the last two items in both questionnaires.

**Order Effect Analysis** The statistical analysis presented in Table 1 shows that order of play does not affect the pairwise preferences of users. The insignificant order effects also, in part, demonstrate that effects such as a user's possible preference for the very first game played and the interplay between reported emotions and familiarity with the game are statistically insignificant. Even though not statistically significant, the correlation statistic values of Table 1 reveal a preference for the second game played for most questionnaire items (negative correlation values).

The H2 hypothesis is ruled-out: no effect exists in any preference questionnaire item while significant effects are observed in the fun, challenge, boredom and relaxation rating questions. These effects may, in part, explain the low  $c(z)$  values in boredom and relaxation but also be responsible for the level of inconsistency in fun and challenge. In general it appears that — excluding the anxiety state —  $c(z)$  values (significant or not) are larger in the rating scheme than in the preference questionnaire scheme. The total order effect is not significant for either questionnaire scheme, which does not allow any safe conclusions to be drawn when all questionnaire items are considered.

The  $r_c$  values in Table 1 demonstrate that both questionnaire schemes are consistent in frustration and excitement and no additional conclusions can be drawn for these two states. On the other hand, it appears as if the inconsistencies of anxiety preferences have

an impact on the low  $c(z)$  value of that state given that the  $r_c$  values are not significant. The order statistics computed including the equal preference (3-AFC) could provide a clearer picture of the relationship between order effects and questionnaire scheme inconsistencies and are left for future analysis due to space considerations.

**Analysis & Conclusions** The statistical analysis for the Maze-Ball case study revealed two main effects: consistency (of varying degree) between rating and preference reports in all 2-AFC questionnaire items and significant order effects for the rating scheme.

Results related to the first effect suggest that even though for some question items (e.g. fun, frustration and challenge) the consistency is higher than others (e.g. anxiety, relaxation and boredom), the hypothesis H1 is ruled out for all emotional states in Maze-Ball. Nevertheless, as in Playware, there are questionnaire items for which the agreement between rating and preferences is far from exact (i.e.  $c(z) = 1.0$ ). For instance, correlation values between 0.6 and 0.7, observed in four out of seven question items of the Maze-ball questionnaire, are significant yet raise questions for the several mismatch instances present in the reports.

The second effect suggests that hypothesis H2 is ruled out. The analysis of order of reporting shows, in general, higher order test statistic values in rating than in preferences and significant order effects in four emotional states when reported via a rating scheme. Both indicate a potential higher degree of randomness reporting with rating schemes for that case study.

Finally, note that the consistency of preferences indicated by the  $r_c$  statistic is more often significant for the 2-AFC answers derived from 4-AFC protocol, which is to be expected since 4-AFC explicitly accounts for cases of non-preference.

## 5 Discussion

This initial set of game case studies and the results obtained raise several questions with respect to the relationship between rating and preference self-reports and the particular game survey studies used to test our hypothesis. While a comparison between the two studies is not appropriate given their large set of dissimilarities, an initial analysis across both test cases will assist the design of additional user survey studies that could shed more light to self-reporting effects.

Most significant is the observation that while direct and derived preferences are generally well-correlated, mismatches occur rather frequently and rating questionnaires appear more susceptible to order-of-play effects than preference questionnaires. It is, therefore, interesting to ask *why reported preferences and ratings do not match exactly?* The two studies presented in this paper link the reporting order effect and the existence of randomness in reporting with the inconsistency between the two self-report schemes. The effect of play order is present in most user states examined. Unsurprisingly, these effects vary across different studies, questionnaire schemes and affective states. In both studies there is a general trend of preference for the second game played (recency or order effect) with significant effects appearing only in the rating scheme. Moreover, the statistic measuring the degree of rating consistency suggests that randomness existent

in rating reports appears to be a critical factor for the inconsistency between the two reporting schemes. Preliminary results of a fairer calculation of the  $r_c$  values — including the equal option of preference and allowing for the equality of rating reports — show that consistency is significant only in the preference scheme, which suggests a benefit of preferences for accurate subjective affective reporting and annotation.

A number of other points are worth noting; a study taking account of all of them exceeds the scope of the present paper, but the results reported here suggest that such a study may be worthwhile.

*The experimental protocol favors expressed rating score.* Rating questions were asked **twice as often** as preference questions were asked. Thus, subjects are expected to be familiar with the structure of the rating scheme more than the preference scheme. The preference scheme is arguably **simpler** for the respondent, but requires **increased short-term memory** since at least two — instead of one in the rating scheme — interaction sessions are necessary for comparison. Moreover, the rating scheme question comes first, straight after the experience, followed by the preference scheme. One would, therefore, expect that cognitive and short-term memory load and furthermore questionnaire completion times would be higher when preferences are reported. However, preliminary results from current game survey studies suggest that the time taken to complete a rating questionnaire is significantly higher than a preference questionnaire.

*Questionnaire usability.* Clearly, usability does not affect the results between the two report schemes since the interaction is the same for both: pen-n-paper in Playware, digital bullet-form questionnaire in Maze-Ball.

*Amount of perceived information.* The amount of information provided through the questionnaire is quite unlikely to have an effect on the findings. All questions, preference or rating, are asked in a similar fashion with very small differences — e.g. “I felt challenged” (rating) vs. “I felt more challenged in:” (preferences). The rating schemes used, however, have more available choice options than the preference schemes. For Playware, the options were 2 for preference and 20 for rating. On the other end, rating and preferences have 5 and 4 options, respectively, for Maze-Ball. The thermometer-like rating scheme of Playware appears to generate higher consistencies between preferences and rating but those consistencies are not apparent in all user expressed states of the Maze-Ball study. The thermometer type of rating questionnaire and the 5-option game experience questionnaire (GEQ) [8] are used for their popularity in user and player experience research. A dedicated control experiment is required to explore the impact of the number of options of the questionnaire schemes on the consistency between expressed rating and preference. Four or three-option rating questionnaires could possibly lead to reduced cognitive load of users and higher consistencies.

*Self-report limitations.* Well known limitations of self-reporting such as **self-deception, high intrusiveness and learnability effects** are applicable to both questionnaire schemes and, thereby, do not seem to have a particular impact on the comparison. While there is no clear way to identify such effects, controlling the order of games and questionnaire sessions, as proposed, alleviates in part such effects inherent in naive questionnaires. Other multimodal input sources, including biofeedback and additional context-based game metrics, could be used for further analysis but do not supplant the self-reports.

## Acknowledgments

The authors would like to thank all subjects that participated in the experiments. Special thanks also goes to Héctor P. Martínez for his help in conducting the Maze-Ball user survey experiment. The research was supported, in part, by the FP7 ICT project SIREN (project no: 258453).

## References

1. Agresti, A.: Analysis of ordinal paired comparison data. *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 41(2), 287–297 (1992)
2. Chan, J.C.: Response-order effects in Likert-type scales. *Educational and Psychological Measurement* 51(3), 531–540 (1991)
3. Cohen, J.: A coefficient of agreement for nominal scales. *Educational and Psychological Measurement* 20, 37–46 (1960)
4. Cronbach, J.L.: Coefficient alpha and the internal structure of tests. *Psychometrika* 16(3), 297–334 (1951)
5. Likert, R.: A technique for the measurement of attitudes. *Archives of Psychology* 140, 1–55 (1932)
6. Linn, R., Gronlund, N.: Measurement and assessment in teaching. Prentice-Hall (2000)
7. Lund, H.H., Klitbo, T., Jessen, C.: Playware technology for physically activating play. *Artificial Life and Robotics Journal* 9(4), 165–174 (2005)
8. Poels, K., IJsselsteijn, W.: Development and validation of the game experience questionnaire. In: FUGA Workshop mini-symposium. Helsinki, Finland (2008)
9. Read, J., MacFarlane, S., Cassey, C.: Endurability, engagement and expectations. In: Proceedings of International Conference for Interaction Design and Children (2002)
10. Scheffe, H.: An analysis of variance for paired comparisons. *Journal of the American Statistical Association* 47(259), 381–400 (1952)
11. Stevens, S.S.: On the Theory of Scales of Measurement. *Science* 103(2684), 677–680 (1946)
12. Tognetti, S., Garbarino, M., Bonarini, A., Matteucci, M.: Modeling enjoyment preference from physiological responses in a car racing game. In: Proceedings of the IEEE Conference on Computational Intelligence and Games. pp. 321–328. Copenhagen, Denmark (18–21 August 2010)
13. Viswanathan, M.: Measurement of individual differences in preference for numerical information. *Journal of Applied Psychology* 78(5), 741–752
14. Yannakakis, G.N.: Preference Learning for Affective Modeling. In: Proceedings of the Int. Conf. on Affective Computing and Intelligent Interaction. pp. 126–131. IEEE, Amsterdam, The Netherlands (September 2009)
15. Yannakakis, G.N., Hallam, J.: Towards Optimizing Entertainment in Computer Games. *Applied Artificial Intelligence* 21, 933–971 (2007)
16. Yannakakis, G.N., Hallam, J., Lund, H.H.: Entertainment Capture through Heart Rate Activity in Physical Interactive Playgrounds. *User Modeling and User-Adapted Interaction, Special Issue: Affective Modeling and Adaptation* 18(1-2), 207–243 (February 2008)
17. Yannakakis, G.N., Maragoudakis, M., Hallam, J.: Preference Learning for Cognitive Modeling: A Case Study on Entertainment Preferences. *IEEE Systems, Man and Cybernetics; Part A: Systems and Humans* 39(6), 1165–1175 (November 2009)
18. Yannakakis, G.N., Martínez, H.P., Jhala, A.: Towards Affective Camera Control in Games. *User Modeling and User-Adapted Interaction* 20(4), 313–340 (2010)

# Ratings are overrated!

Georgios N. Yannakakis\* and Héctor P. Martínez

Institute of Digital Games, University of Malta, Msida, Malta

## OPEN ACCESS

**Edited by:**

Javier Jaen,  
Universitat Politecnica de Valencia,  
Spain

**Reviewed by:**

Andreas Duenser,  
Commonwealth Scientific and  
Industrial Research Organisation,  
Australia

Eran Toch,  
Tel Aviv University, Israel  
Donald Glowinski,  
University of Geneva, Switzerland

**\*Correspondence:**

Georgios N. Yannakakis,  
Institute of Digital Games, University  
of Malta, Msida 2080, Malta  
georgios.yannakakis@um.edu.mt

**Specialty section:**

This article was submitted to  
Human-Media Interaction, a section  
of the journal Frontiers in ICT

**Received:** 01 April 2015

**Accepted:** 09 July 2015

**Published:** 30 July 2015

**Citation:**

Yannakakis GN and Martínez HP  
(2015) Ratings are overrated!  
Front. ICT 2:13.  
doi: 10.3389/fict.2015.00013

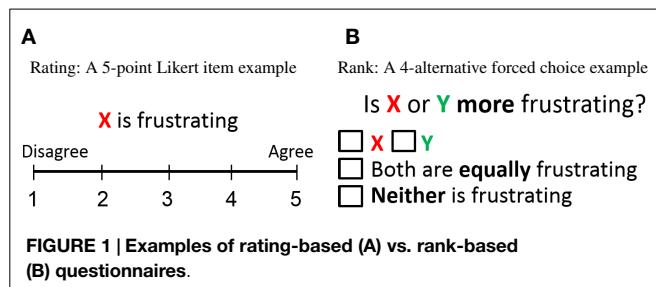
Are ratings of any use in human–computer interaction and user studies at large? If ratings are of limited use, is there a better alternative for quantitative subjective assessment? Beyond the intrinsic shortcomings of human reporting, there are a number of supplementary limitations and fundamental methodological flaws associated with *rating-based questionnaires* – i.e., questionnaires that ask participants to rate their level of agreement with a given statement, such as a Likert item. While the effect of these pitfalls has been largely downplayed, recent findings from diverse areas of study question the reliability of using ratings. *Rank-based questionnaires* – i.e., questionnaires that ask participants to rank two or more options – appear as the evident alternative that not only eliminates the core limitations of ratings but also simplifies the use of sound methodologies that yield more reliable models of the underlying reported construct: user emotion, preference, or opinion. This paper solicits recent findings from various disciplines interlinked with psychometrics and offers a quick guide for the use, processing, and analysis of rank-based questionnaires for the unique advantages they offer. The paper challenges the traditional state-of-practice in human–computer interaction and psychometrics directly contributing toward a paradigm shift in subjective reporting.

**Keywords:** ratings, Likert-scale, ranks, psychometrics, subjective reporting, questionnaires

## Introduction

The key research question within psychometrics and user studies is how to best approximate a user's notion of a subjective construct, such as an experience, a cognitive state, an emotion, or a preference. Even though the *ground truth* of a user's internal state can be manifested via numerous cognitive processes or bodily alterations, it is still far from trivial how to best assess and process those manifestations; entire research areas, such as user experience, user modeling, and affective computing, are long dedicated to this task. Although subjective reporting (first- or third-person) comes with several limitations, such as self-deception and memory-biases, it offers the most direct and popular approach to the annotation of subjective constructs. Thus, quantitative reports via questionnaires provide unique properties for evaluating the capacity of interactive systems (Bardram et al., 2013; Chen et al., 2014) and for constructing computational models of reported user states (Hernandez et al., 2014).

The dominant practice within *human–computer interaction (HCI)* for quantitatively assessing aspects of a user's behavior, experience, opinion, or emotion relies on subjective assessment via rating-based questionnaires – see Bardram et al. (2013), Bryan et al. (2014), Chen et al. (2014), Goyal et al. (2014), Hernandez et al. (2014), Mauderer et al. (2014), Schild et al. (2014), and Sonderegger et al. (2014) among many. Indicatively, a thorough analysis of the papers published in the most prestigious HCI conference last year (Proceedings of CHI'14) reveals that the majority of accepted papers use some form of quantitative assessment approach and more than 80% of these rely on rating-based questionnaires. Popular rating-based questionnaires (see **Figure 1A** for an example) include the Likert-scale (Likert, 1932), the Geneva Wheel model (Scherer, 2005), the Self-Assessment



Manikin (Morris, 1995), the Positive and Negative Affect Schedule (Sonderegger et al., 2014), and the Game Experience Questionnaire (IJsselsteijn et al., 2008). The obtained answers are either used as a means to evaluate an interactive system via the experience of its users – see Bryan et al. (2014), Chen et al. (2014), and Mauderer et al. (2014) – or as data for building predictive models of user reports – i.e., user modeling (Martínez et al., 2014; Hernandez et al., 2014). On the other hand, rank-based questionnaires – which ask the participant to rank a preference between two (or among more than two) options – still remain a rarely used instrument of subjective assessment and modeling, even though there is already significant evidence for their advantages over rating-based questionnaires (Yannakakis and Hallam, 2011; Metallinou and Narayanan, 2013; Čopić Pucihar et al., 2014). An example of a rank-based questionnaire (4-alternative forced choice) is illustrated in **Figure 1B**.

This paper contributes toward a *shift* of the current state-of-practice in user experience, HCI, and psychometrics research at large. For that purpose, the paper provides clear evidence that rating-based evaluation (and modeling) is detrimental to psychometrics and HCI research efforts as it points to biased representations of a user's subjective report. As a result, rating-based instruments are not only of questionable use for the analysis of a subject's report but also evidently lead to unreliable models of those subjects and their reports.

The paper is novel in that it collectively solicits empirical evidence from various research fields, such as marketing research, applied statistics, affective computing, user modeling, and user experience to draw the multiple advantages of rank-based questionnaires for psychometrics and HCI research. At the same time, it provides a comprehensive guide on the use, processing, and analysis of rank-based questionnaires. Toward that aim, we object the use of ratings for HCI based on a number of fundamental limitations and practice flaws (see next section) and we provide empirical evidence for the advantages of ranks (compared to ratings) with respect to subjectivity, order, and inconsistency effects. Furthermore, we suggest appropriate data processing techniques on how to treat ratings – when those are available – and we introduce an open-source toolbox that supports those techniques.

## Ratings: Limitations and Fundamental Flaws

The vast majority of user and psychometric studies have adopted rating questionnaires to capture the opinions, preferences, and perceived experiences of experiment participants – see Bryan et al. (2014), Chen et al. (2014), and Mauderer et al. (2014) among

many. The most popular rating-based questionnaire follows the principles of a Likert-scale (Likert, 1932) in which users are asked to specify their level of agreement with (or disagreement against) a given statement. Ratings have been used, for instance, to report the level of comfort and ease of use of new interfaces or devices (Chen et al., 2014; Weigel et al., 2014) or the stress level during a given task – e.g., in Bardram et al. (2013) and Hernandez et al. (2014). Rating-based reporting, however, has notable inherent limitations that are often overlooked, resulting in fundamentally flawed analyses (Jamieson, 2004). This section sheds some light on the most critical of these limitations and flaws.

### Inherent Limitation: Inter-Personal Differences

Traditionally, HCI studies analyze ratings by comparing their values across participants – see Goyal et al. (2014) and Mark et al. (2014) among many. This is a generally accepted and dominant practice in the community but it neglects the existence of inter-personal differences on the rating process as the meaning of each level on a rating scale may differ across experiment participants. For example, two participants may assess the exact same level of “ease to use” for a new device but then one rates it as “very easy to use” and the other as “extremely easy to use.” There are numerous factors that contribute to the different internal rating scales existent across participants (Metallinou and Narayanan, 2013), such as differences in personality, culture (Sneddon et al., 2011), temperament, and interests (Viswanathan, 1993). As these factors are documented extensively in the literature, the appropriateness of the dominant HCI state-of-practice is directly questioned.

A large volume of studies have also identified the presence of primacy and recency order effects in rating-based questionnaires e.g., Chan (1991) and Yannakakis and Hallam (2011), seen as systematic biases toward parts of the scale (Linn and Gronlund, 2000) (e.g., right handed participants may tend to use the right side of the scale) or a fixed tendency over time (e.g., on a series of experimental conditions, the last ones are rated higher). Indicatively, the comparative study of Yannakakis and Hallam (2011) between ratings and ranks showcases higher inconsistency effects and significant order (recency) effects existent in ratings across two different datasets, which contain both rank and rating annotations obtained from the same participants. Although these are systematic biases (opposed to personal), they pose additional challenges on the comparison of ratings among participants, as participants are affected to different extents. Even though experiments on quantifying human perception through rating questionnaires have led to interesting findings on the relationship between perception and reporting, biases of the use of ratings as an assessment tool have not been examined (Jay et al., 2007).

### Ratings are Not Numbers

In addition to inter-personal differences, a critical limitation arises when ratings are treated as interval values since ratings are by nature ordinal values (Stevens, 1946; Jamieson, 2004). As a result, any method that treats them as numbers (e.g., average values, *t*-tests, linear models) is fundamentally flawed. In most questionnaires, Likert items are represented as pictures [e.g., different representations of arousal in the Self-Assessment Manikin (Morris, 1995)] or as adjectives (e.g., “moderately,” “fairly,” and “extremely”). These labels (images or adjectives) are

often erroneously converted to integer numbers violating basic axioms of statistics, which suggest that ordinal values cannot be treated as interval values (Stevens, 1946) since the underlying numerical scale is unknown. Note that even when a questionnaire features ratings as numbers (e.g., see **Figure 1A**), the scale is still ordinal as the numbers in the instrument are only labels; thus, the underlying numerical scale is still unknown and dependent on the participant (Stevens, 1946; Langley and Sheppeard, 1985; Ovadia, 2004). Moreover, when treated as numbers, equal ratings are considered of equal value. This is another invalid assumption to make as questionnaires do not always provide sufficient granularity. By treating ratings as ordinal values, this issue is avoided as only the relations among unequal values are considered.

## The Non-Linearity of Ratings

Treating ratings as interval values is grounded in the assumption that the difference between consecutive ratings is fixed (i.e., ratings follow a linear scale). However, there is no valid assumption suggesting that a subjective rating scale is linear (Jamieson, 2004). For instance, the difference between “fairly (4)” and “extremely (5)” may be larger than the distance between “moderately (3)” and “fairly (4)” as some experiment participants rarely use the extremes of the scale or tend to use one extreme more than the other (Langley and Sheppeard, 1985). If, instead, ratings are treated naturally as ordinal data no assumptions are made about the distance between rating labels, which eliminates introducing flawed information and data noise to the analysis.

## Why Should I Use Ranks Instead?

A rank-based questionnaire scheme asks experiment participants to compare and sort a number of options. On its simplest form, the participants compare two options and specify which one is the preferred under a given statement (pairwise preference). For instance, participants could select which of two devices is easier to use. With more than two options, the participants are asked to provide a ranking of some or all the options. At a remote observation, one may argue that ranks provide less information than ratings as they do not express a quantity explicitly and only provide ordinal relations. As argued in the previous section, however, any additional information obtained by ratings when treated as numbers violates basic axioms of applied statistics. Thus, ratings do not provide data for a richer analysis if appropriately treated as ordinal values.

Being a form of subjective reporting rank-based questionnaires (as much as rating-based questionnaires) is associated with well known limitations, such as memory effects and self-deception. Reporting about subjective constructs, such as experience, preference, or emotion via rank-based questionnaires, however, has recently attracted the interest of researchers in marketing (Dhar and Simonson, 2003), psychology (Brown and Maydeu-Olivares, 2013), user modeling (Yang and Chen, 2011; Baveye et al., 2013), and affective computing (Tognetti et al., 2010; Martínez et al., 2014) among other fields. This gradual paradigm shift is driven by both the reported benefits of ranks minimizing the effects of self-reporting subjectivity and recent findings demonstrating the advantages of ranks over ratings. Inspired by the seminal work of Scheffe (1952) and Agresti (1992) for

the analysis of paired comparisons Yannakakis and Hallam (2011) compared data from rating and rank-based questionnaires across a number of domains and identified increased order and inconsistency effects when ratings are used. Evidence from findings by Metallinou and Narayanan (2013) also suggest that rank-based annotation of emotion should be preferred to rating-based annotation for its ability to eliminate annotation biases (cultural, subjective, inconsistency, inter-rater, etc.).

In summary, results across different domains investigating subjective assessment suggest that rank-based reports minimize the assumptions made about experiment participants’ notions of highly subjective constructs, such as experience and emotions, and allow a fair comparison among the answers of different participants. Moreover, artifacts, such as the subjective notion of scaling, are eliminated. Finally, all these advantages also lead to the construction of generalizable and accurate computational models of users or their experience (Martínez et al., 2014).

## What if Ratings is All I Have?

The core findings from the areas of applied statistics, user modeling, affective computing, machine learning, and marketing research discussed already not only suggest that ranks define a superior instrument for subjective assessment but they also *question the very use* of ratings at the first place. One could, however, still claim that the use of ratings in some particular experimental protocols is unavoidable. For instance, in experimental protocols, subjects can only be asked to assess their experience on solely one version of an interactive system (e.g., a game, a web-browser).

When faced with such a condition ratings could provide a viable assessment instrument if they are naturally treated as ordinal data. A recent study by Martínez et al. (2014) investigates the effect of using ratings as nominal or ordinal scales when studying the relation between physiological attributes and emotional states. Both approaches are tested on synthetic (testing “*in vitro*”) and human (testing “*in vivo*”) ratings. The core findings of the study across all datasets examined provide clear evidence that ratings (when used) should be naturally transformed to ordinal representations (ranks). This practice has clear benefits: any data analysis followed yields more reliable and generalizable outcomes as those better approximate the underlying *ground truth* of the reported subjective construct. The transformation from ratings to ranks is straightforward. Ratings are compared to one another and a pairwise preference/ranking is created for every pair/tuple; higher ratings take the top positions of the ranking and lower ratings the positions in the bottom of the ranking. Comparisons of ratings from different experiment participants must be avoided. In addition, if the time window between reported ratings is sufficiently large for the examined task (e.g., in the magnitude of hours or more) one can also consider removing particular rating pairs to reduce artifacts connected to participants’ episodic memory.

In general, approaches for analyzing ordinal ratings should rely on *non-parametric* statistics: from simple statistical explorations via *Spearman’s* correlation, to significance tests via the *Mann–Whitney* test and the *Wilcoxon signed-rank* test for paired samples (Wilcoxon, 1945), to the *Kruskal–Wallis* (Kruskal and Wallis, 1952) and *Friedman’s* (Friedman, 1940) tests for three (or more) groups of ranks. Clearly, statistical models, such as

artificial neural networks and support vector machines, are also suitable for the analysis of ordinal data (Martínez et al., 2014).

Norman (2010) showed empirically in one dataset that *Pearson's* correlation (which treats ratings as intervals) is robust enough when compared against *Spearman's* rank correlation (which treats ratings as ordinal values). Such evidence could support the validity of using standard *parametric* correlation tests that treat ratings as interval values but does not question the very use of ratings due to their inherent limitations. On the contrary, a number of studies have demonstrated the supremacy of ranks in eliminating various forms of reporting biases [e.g., Yannakakis and Hallam (2011)]. Finally, significant improvements have been reported in accuracies of non-linear statistical models when ratings are treated as ordinal values (Martínez et al., 2014).

## How to Analyze Ranks

Standard data visualization methods based on averages or SDs are strictly not applicable on ordinal data – obtained directly as ranks or transformed from ratings. Instead, to explore the relationships between ranks and a number of considered factors, a stacked bar chart can be used to visualize how many observations were assigned to each rank for each value of the factor.

For a statistical factor analysis, a common choice is the *Wilcoxon signed-rank test* (Wilcoxon, 1945), sometimes also used to evaluate the effect of ratings as, for instance, in Mauderer et al. (2014). This is a paired-samples test and, therefore, guarantees that only within-participant ranks are compared, bypassing inter-personal differences. A common alternative is *Kendall's Tau* (Kendall, 1938) that can be used to calculate the correlation between the hypothesized order (e.g., device A is easier to use than device B) and the observed ranks – see, e.g., Martínez et al. (2014). The non-parametric *Kruskal–Wallis* and *Friedman's* tests mentioned earlier are also applicable.

Furthermore, if an HCI researcher is interested in using the reported ranks to build computational models that predict those ranks (e.g., constructing models of users) a large palette of algorithms is currently available. Linear statistical models, such as linear discriminant analysis and large margins, and non-linear approaches, such as Gaussian processes, artificial neural networks, and support vector machines, are applicable for learning to predict ranks. These methods are derived from the sub-area of machine learning named *preference learning* (Fürnkranz and Hüllermeier, 2010). A number of such preference learning methods as well as data preprocessing and feature selection algorithms are currently included in the *preference learning toolbox (PLT)* (Farrugia et al., 2015). PLT is an open-access, user-friendly, and accessible toolkit<sup>1</sup> built and constantly updated for the purpose of easing the processing of (and promoting the use of) ranks.

## Summary of Conclusions

This paper directly objects to the use and analysis of subjective assessment via ratings within quantitative user studies, HCI and psychometrics research contributing to a shift from the dominant state of practice in those fields. Beyond any well-reported

limitations of subjective reporting (e.g., memory effects, self-deception) ratings come with inherited limitations as an instrument of reporting. These are derived from inter-personal differences and include, among many, high-inconsistency effects and subjectivity of scaling. Those effects question the very use of ratings for obtaining valid data for any further analysis. Most importantly, the traditional analysis of ratings within HCI and psychometrics – i.e., deriving statistical properties from ratings, such as average and variance values – violates two fundamental assumptions. The first common flaw is the violation of the assumption that ratings are ordinal data. The second assumption violated is that ratings evidently are not linear (even if they could be represented as numbers). In response to the above mathematical violations, in principle, ratings *should not* be converted to numerical scales and analyzed as numbers.

Rank-based questionnaires are the alternative instrument for subjective quantitative assessment proposed in this paper. Recent findings from a number of fields including applied statistics, affective computing, user modeling, and machine learning provide clear evidence for the supremacy of ranks, when compared to ratings, on minimizing subjectivity biases as well as order, inter-rater, and inconsistency effects (Yannakakis and Hallam, 2011; Metallinou and Narayanan, 2013). More so, recent evidence suggests that we can construct more accurate and reliable statistical models of reported ratings – that better approximate the underlying ground truth of the subjective construct we attempt to measure – only when ratings are naturally treated as ordinal data (Martínez et al., 2014).

Given the supremacy or rank-based subjective assessment for both evaluating interactive systems (through the experience of their users) and as the ground truth for deriving computational models of subjective reports, this paper serves as a guide for both rank-based evaluation and rank-based computational modeling. For the former, it provides methods for the conversion of ratings to ranks – when ratings are available – and an overview of statistical processes for rank reports. For the latter, it proposes preference learning methods and algorithms – incorporated to an open-source, accessible toolbox – for the construction of predictive models of ranks.

It is important to stress that this paper did not intend to present *yet another* case study to further prove empirically the advantages of ranks over ratings or demonstrate the general flaws of processing ratings. Our claims are not based on the popularity of ranks in other fields outside HCI, but on empirical findings as surveyed in the paper. While the limitations of ratings and ranks have been identified and discussed extensively, no other study within the HCI community both solicits evidence for the comparative advantages of ranks (as demonstrated in other fields) and offers a short guidebook on how to process ranks statistically. We hope that this paper highlights the obvious fundamental issues of ratings as a subjective assessment tool and introduces ranks as the alternative reporting approach toward altering a dominant, yet falsified, community practice.

## Acknowledgments

The work is supported, in part, by the EU-funded FP7 ICT ILearnRW project (project no: 318803).

<sup>1</sup><http://sourceforge.net/projects/pl-toolbox/>

## References

- Agresti, A. (1992). Analysis of ordinal paired comparison data. *J. R. Stat. Soc. Ser. C Appl. Stat.*, 41, 287–297.
- Bardram, J. E., Frost, M., Szántó, K., Faurholt-Jepsen, M., Vinberg, M., and Kessing, L. V. (2013). Designing mobile health technology for bipolar disorder: a field trial of the monarca system. *Proc. SIGCHI Conf. Hum. Factor. Comput. Syst.* 2627–2636. doi:10.1145/2470654.2481364
- Baveye, Y., Bettinelli, J. N., Dellandrea, E., Chen, L., and Chamaret, C. (2013). A large video database for computational models of induced emotion. *Proc. of Affect. Comput. Intell. Int.* 13–18. doi:10.1109/ACII.2013.9
- Brown, A., and Maydeu-Olivares, A. (2013). How irt can solve problems of ipsative data in forced-choice questionnaires. *Psychol. Methods* 18, 36. doi:10.1037/a0030641
- Bryan, N. J., Mysore, G. J., and Wang, G. (2014). Isse: an interactive source separation editor. *Proc. SIGCHI Conf. Hum. Factor. Comput. Syst.* 257–266. doi:10.1145/2556288.2557253
- Chan, J. C. (1991). Response-order effects in Likert-type scales. *Educ. Psychol. Meas.* 51, 531–540. doi:10.1177/0013164491513002
- Chen, X. A., Grossman, T., Wigdor, D. J., and Fitzmaurice, G. (2014). Duet: exploring joint interactions on a smart phone and a smart watch. *Proc. SIGCHI Conf. Hum. Factor. Comput. Syst.* 159–168. doi:10.1145/2556288.2556955
- Čopić Pucihar, K., Coulton, P., and Alexander, J. (2014). The use of surrounding visual context in handheld ar: device vs. user perspective rendering. *Proc. SIGCHI Conf. Hum. Factor. Comput. Syst.* 197–206. doi:10.1145/2556288.2557125
- Dhar, R., and Simonson, I. (2003). The effect of forced choice on choice. *J. Market. Res.* 40, 146–160. doi:10.1509/jmkr.40.2.146.19229
- Farrugia, V. E., Martínez, H. P., and Yannakakis, G. N. (2015). The preference learning toolbox. [arXiv:1506.01709].
- Friedman, M. (1940). A comparison of alternative tests of significance for the problem of m rankings. *Ann. Math. Stat.* 11, 86–92. doi:10.1214/aoms/1177731944
- Fürnkranz, J., and Hüllermeier, E. (2010). *Preference Learning*. New York: Springer.
- Goyal, N., Leshed, G., Cosley, D., and Fussell, S. R. (2014). Effects of implicit sharing in collaborative analysis. *Proc. SIGCHI Conf. Hum. Factor. Comput. Syst.* 129–138. doi:10.1145/2556288.2557229
- Hernandez, J., Paredes, P., Roseway, A., and Czerwinski, M. (2014). Under pressure: sensing stress of computer users. *Proc. SIGCHI Conf. Hum. Factor. Comput. Syst.* 51–60. doi:10.1145/2556288.2557165
- IJsselsteijn, W., Poels, K., and De Kort, Y. (2008). *The Game Experience Questionnaire: Development of a Self-Report Measure to Assess Player Experiences of Digital Games*. Eindhoven: TU Eindhoven.
- Jamieson, S. (2004). Likert scales: how to (ab) use them. *Med. Educ.* 38, 1217–1218. doi:10.1111/j.1365-2929.2004.02012.x
- Jay, C., Glencross, M., and Hubbeld, R. (2007). Modeling the effects of delayed haptic and visual feedback in a collaborative virtual environment. *ACM Trans. Comput. Hum. Interact.* 14, 1275514. doi:10.1145/1275511
- Kendall, M. G. (1938). A new measure of rank correlation. *Biometrika* 81–93. doi:10.1093/biomet/30.1-2.81
- Kruskal, W. H., and Wallis, W. A. (1952). Use of ranks in one-criterion variance analysis. *J. Am. Stat. Assoc.* 47, 583–621. doi:10.1080/01621459.1952.10483441
- Langley, G., and Shepphard, H. (1985). The visual analogue scale: its use in pain measurement. *Rheumatol. Int.* 5, 145–148. doi:10.1007/BF00541514
- Likert, R. (1932). A technique for the measurement of attitudes. *Arch. Psychol.* 14, 1–55.
- Linn, R., and Gronlund, N. (2000). *Measurement and Assessment in Teaching*. Upper Saddle River, NJ: Prentice-Hall.
- Mark, G., Wang, Y., and Niiya, M. (2014). Stress and multitasking in everyday college life: an empirical study of online activity. *Proc. SIGCHI Conf. Hum. Factor. Comput. Syst.* 41–50. doi:10.1145/2556288.2557361
- Martínez, H. P., Yannakakis, G. N., and Hallam, J. (2014). Don't classify ratings of affect; rank them! *IEEE Trans. Affect. Comput.* 5, 314–326. doi:10.1109/TAFFC.2014.2352268
- Mauderer, M., Conte, S., Nacenta, M. A., and Vishwanath, D. (2014). Depth perception with gaze-contingent depth of field. *Proc. SIGCHI Conf. Hum. Factor. Comput. Syst.* 217–226. doi:10.1145/2556288.2557089
- Metallinou, A., and Narayanan, S. (2013). “Annotation and processing of continuous emotional attributes: challenges and opportunities,” in *Automatic Face and Gesture Recognition (FG), 2013 10th IEEE International Conference and Workshops on* (Shanghai: IEEE), 1–8.
- Morris, J. (1995). Observations: sam: the self-assessment Manikinan efficient cross-cultural measurement of emotional response. *J. Advert. Res.* 35, 63–68.
- Norman, G. (2010). Likert scales, levels of measurement and the laws of statistics. *Adv. Health Sci. Educ.* 15, 625–632. doi:10.1007/s10459-010-9222-y
- Ovadia, S. (2004). Ratings and rankings: reconsidering the structure of values and their measurement. *Int. J. Soc. Res. Method.* 7, 403–414. doi:10.1080/1364557032000081654
- Scheffe, H. (1952). An analysis of variance for paired comparisons. *J. Am. Stat. Assoc.* 47, 381–400. doi:10.2307/2281310
- Scherer, K. (2005). What are emotions? and how can they be measured? *Soc. Sci. Inform.* 44, 695–729. doi:10.1177/0539018405058216
- Schild, J., La Viola, J. J. Jr., and Masuch, M. (2014). Altering gameplay behavior using stereoscopic 3d vision-based video game design. *Proc. SIGCHI Conf. Hum. Factor. Comput. Syst.* 207–216. doi:10.1145/2556288.2557283
- Sneddon, I., McKeown, G., McRorie, M., and Vukicevic, T. (2011). Cross-cultural patterns in dynamic ratings of positive and negative natural emotional behaviour. *PLoS ONE* 6:e14679. doi:10.1371/journal.pone.0014679
- Sonderegger, A., Uebelbacher, A., Pugliese, M., and Sauer, J. (2014). “The influence of aesthetics in usability testing: the case of dual-domain products,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY: ACM), 21–30.
- Stevens, S. S. (1946). On the theory of scales of measurement. *Science* 103, 677–680. doi:10.1126/science.103.2684.677
- Tognetti, S., Garbarino, M., Bonarini, A., and Matteucci, M. (2010). “Modeling enjoyment preference from physiological responses in a car racing game,” *Computational Intelligence and Games (CIG), 2010 IEEE Symposium on* (Copenhagen: IEEE), 321–328.
- Viswanathan, M. (1993). Measurement of individual differences in preference for numerical information. *J. Appl. Psychol.* 78, 741–752. doi:10.1037/0021-9010.78.5.741
- Weigel, M., Mehta, V., and Steimle, J. (2014). More than touch: understanding how people use skin as an input surface for mobile computing. *Proc. SIGCHI Conf. Hum. Factor. Comput. Syst.* 179–188. doi:10.1145/2556288.2557239
- Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics Bull.* 80–83. doi:10.2307/3001968
- Yang, Y. H., and Chen, H. H. (2011). Ranking-based emotion recognition for music organization and retrieval. *IEEE Trans. Audio Speech Lang. Process.* 19, 762–774. doi:10.1109/TASL.2010.2064164
- Yannakakis, G. N., and Hallam, J. (2011). Rating vs. preference: a comparative study of self-reporting. *Proc. Affect. Comput. Intell. Int.* 6974, 437–446. doi:10.1007/978-3-642-24600-5\_47
- Conflict of Interest Statement:** The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.
- Copyright © 2015 Yannakakis and Martínez. This is an open-access article distributed under the terms of the Creative Commons Attribution License (CC BY). The use, distribution or reproduction in other forums is permitted, provided the original author(s) or licensor are credited and that the original publication in this journal is cited, in accordance with accepted academic practice. No use, distribution or reproduction is permitted which does not comply with these terms.*

# Dynamic Difficulty Adjustment for Maximized Engagement in Digital Games

Su Xue  
Electronic Arts, Inc.  
209 Redwood Shores Pkwy  
Redwood City, CA, USA  
[sxue@ea.com](mailto:sxue@ea.com)

Navid Aghdaie  
Electronic Arts, Inc.  
209 Redwood Shores Pkwy  
Redwood City, CA, USA  
[naghdaie@ea.com](mailto:naghdaie@ea.com)

Meng Wu<sup>\*</sup>  
Electronic Arts, Inc.  
209 Redwood Shores Pkwy  
Redwood City, CA, USA  
[mewu@ea.com](mailto:mewu@ea.com)

Kazi A. Zaman  
Electronic Arts, Inc.  
209 Redwood Shores Pkwy  
Redwood City, CA, USA  
[kzaman@ea.com](mailto:kzaman@ea.com)

John Kolen  
Electronic Arts, Inc.  
209 Redwood Shores Pkwy  
Redwood City, CA, USA  
[jkolen@ea.com](mailto:jkolen@ea.com)

## ABSTRACT

Dynamic difficulty adjustment (DDA) is a technique for adaptively changing a game to make it easier or harder. A common paradigm to achieve DDA is through heuristic prediction and intervention, adjusting game difficulty once undesirable player states (e.g., boredom or frustration) are observed. Without quantitative objectives, it is impossible to optimize the strength of intervention and achieve the best effectiveness.

In this paper, we propose a DDA framework with a global optimization objective of maximizing a player's engagement throughout the entire game. Using level-based games as our example, we model a player's progression as a probabilistic graph. Dynamic difficulty reduces to optimizing transition probabilities to maximize a player's stay time in the progression graph. We have successfully developed a system that applies this technique in multiple games by Electronic Arts, Inc., and have observed up to 9% improvement in player engagement with a neutral impact on monetization.

## Keywords

Dynamic difficulty adjustment, player engagement optimization, progression model

## 1. INTRODUCTION

Difficulty is a critical factor in computer games and is a challenging factor to set appropriately. Game developers often use pre-defined curves that manipulate the level difficulty as players advance. Although these difficulty curves

\*The first two authors contributed equally to this paper.

©2017 International World Wide Web Conference Committee (IW3C2), published under Creative Commons CC BY 4.0 License. WWW 2017, April 3–7, 2017, Perth, Australia. ACM 978-1-4503-4913-0/17/04. <http://dx.doi.org/10.1145/3041021.3054170>



are usually defined by experienced designers with strong domain knowledge, they have many problems. First, the diversity among players is large. Players have a wide variety of experiences, skills, learning rates, and playing styles, and will react differently to the same difficulty setting. Second, even for an individual player, one's difficulty preference may also change over time. For example, in a level progression game, a player who loses the first several attempts to one level might feel much less frustrated compared to losing after tens of unsuccessful trials.

In contrast to static difficulty, dynamic difficulty adjustment (DDA) addresses these concerns. Such methods exhibit diversity in the levers that adjust difficulty, but share a common theme: prediction and intervention. DDA predicts a player's future state given current difficulty, and then intervenes if that state is undesirable. The strength of this intervention, however, is both heuristic and greedy. The adjustment might be in the right direction, such as making a level easier for a frustrated player. But how easy should the game be to achieve optimal long term benefit is an open question.

In this paper, we will address these issues by defining dynamic difficulty adjustment within an optimization framework. The global objective within this framework is to maximize a player's engagement throughout the entire game. We first model a player's in-game progression as a probabilistic graph consisting of various player states. When progressing in the game, players move from one state to another. The transition probabilities between states are dependent on game difficulties at these states. From this perspective, maximizing a player's engagement is equivalent to maximizing the number of transitions in the progression graph. This objective reduces to a function of game difficulties at various states solvable by dynamic programming. While we focus on level-based games as the context of this presentation, our DDA framework is generic and can be extended to other genres.

The proposed technique has been successfully deployed by Electronic Arts, Inc (EA). We developed a DDA system within the EA Digital Platform, to which game clients request and receive dynamic difficulty advice in realtime. With A/B experiments, we have observed significant im-

creases in core player engagement metrics while seeing neutral impact on monetization. Last, but not least, our DDA recommendations are also used by game designers to refine the game design. For example, when our service repeatedly recommends easier difficulty for a certain level, the game designer knows to decrease the pre-defined difficulty of that level to satisfy the majority of population.

To sum up, the core contributions of this paper are:

- We propose a DDA framework that maximizes a player’s engagement throughout the entire game.
- We introduce a probabilistic graph that models a player’s in-game progression. With the graph model, we develop an efficient dynamic programming solution to maximize player engagement.
- We describe a real-time DDA system that successfully boosted engagement of multiple mobile games.

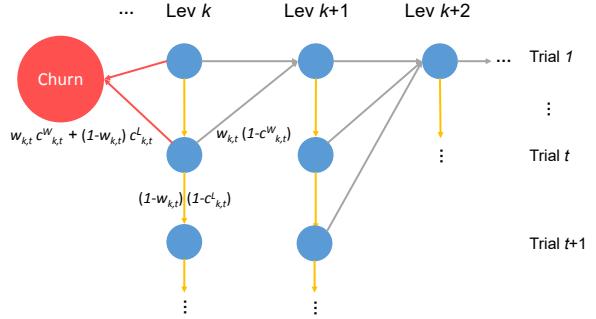
In the remainder of this paper, we will first review related DDA research. We then introduce the graph model of player’s progression and describe the objective function and our optimization solution. We will next report on the application of this DDA technique in a live game as a case study. Finally, we discuss the results of this case study and our future directions.

## 2. RELATED WORK

Personalized gaming is one of the major trends for digital interactive games in recent years [10]. Personalization approaches include content generation, personalized gameplay, and dynamic difficulty adjustment. The theme shared by almost all game difficulty adjustment studies is that they attempt to prevent a player from transitioning to undesired states, such as boredom or frustration. There are several common challenges in difficulty adjustment. For example, how to evaluate a player’s current state? How to predict a player’s upcoming state? What levers are appropriate to use? How to adjust the levers to most appropriate difficulty level? In this section, we review how previous work addressed these questions from different perspectives.

Many approaches are based on the evaluation of players’ skill and performance, and then adapting game difficulty to match the skill level. Zook et al. conducted a series of investigations following this idea [15, 16]. They proposed a data-driven predictive model that accounts for temporal changes in player skills. This predictive model provides a guide for just-in-time procedural content generation and achieves dynamic difficulty adjustment. Similarly, Jennings et al. automatically generate 2D platformer levels in a procedural way [9]. They developed a simulator where players play a short segment of a game for data collection. From this data, they constructed a statistical model of the level element difficulty. They also learned player skill model from the simulator data. Hagelback et al. [6] studied dynamic difficulty by measuring player experience in Real Times Strategy (RTS) games. They, too, use an experimental gaming environment to evaluate testing players’ subjective enjoyment according to different dynamic difficulty schemes.

The majority of DDA systems rely upon prediction and intervention as their fundamental strategy. Hamlet is a well known AI system using Valve’s *Half Life* game engine [8].



**Figure 1: A player’s progression model in a typical level-based game.** We use a probabilistic graph consisting of player states (each circles) to model this progression. Two dimensions, levels and trials are used to identify different states. The directional edges represent possible transition between these states.

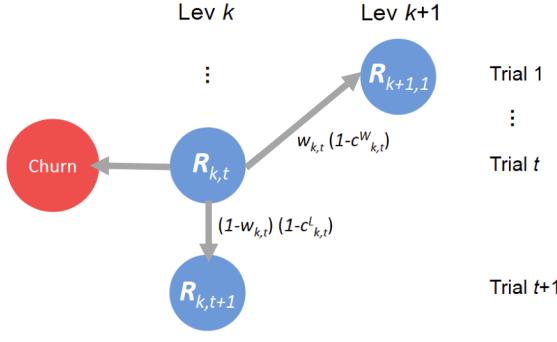
It takes advantages of the flow model developed by Csikszentmihalyi [3], that defines player states on two dimensions: skill and challenge. They suggested the game challenge should match the player skill, therefore some states are preferable while others are not. Hamlet predicts player states, and adjusts the game to prevent inventory shortfalls. Missura and Gartner [11] formalizes the prediction in a more rigorous probabilistic framework. They try to predict the “right” difficulty by formulating the task as an online learning problem on partially ordered sets. Hawkins et al. [7] takes players’ risk profile into account when predicting player performance. Cautious players and risk takers also behave differently in response to dynamic balancing mechanics.

Researchers have also explored the use of various levers to achieve dynamic difficulty. It is preferred that the adjustment by the levers be invisible to players so that they do not feel coddled or abused. Popular levers in the DDA literature include procedural level content [9, 15, 16], off stage elements (such as weapon or support inventory) [8], and AI assistant or opponent [4, 13].

Almost all previous work shares a **common limitation**. These approaches focus on **short-term effects**, i.e., using the difficulty adjustment to immediately rescue player from undesired states. With the prediction and intervention strategy, these methods tend to perform **greedy actions**, often **leading to short-term benefits, but failing to achieve long-term optimal impacts**. In contrast, our proposed technique achieves DDA by maximizing a long-term objective, such as player’s engagement throughout the entire game. In the following section, we describe how to model the entire game engagement, achieve global optimality, and keep players in the game.

## 3. PLAYER PROGRESSION MODEL

We focus on level-based games in this paper. In a level-based game, a player can unlock and advance to the higher levels only if the player wins the current level. There are many well known digital games e.g. Super Mario Bros. (Nintendo Co., Ltd.) and Plants vs. Zombies (Electronic Arts, Inc.) belong to the level-based game category. We first introduce a progression graph to model players’ progression



**Figure 2: A zoom-in look of the state transitions and the associated rewards.** The reward at  $s_{k,t}$ , i.e.,  $R_{k,t}$ , is the weighted sum of the awards at the adjacent states. This property leads to reward maximization with dynamic programming.

trajectories in the game. The modeling approach described below can be generalized to other game types as well.

Defining appropriate states is the key to constructing a progression model. Specifically for level-based games, we simply define the player progression state with two dimensions, level and trial. A player can either advance to a higher level or remain on the current level with repeated trials. Fig. 1 illustrates the progression graph schema. We denote state that a player is playing the  $k$ -th level at the  $t$ -th trial as  $s_{k,t}$ . Within the progression graph, a player’ progression can be represented by a sequence of transitions between states. For example, when a player completes one level, he will advance to a new first trial state in the next level. When a player fails and retries the same level, he will move to the next trial state on the same level. A special, but critical, state is the churn state. Players who enter the churn state will never return to the game. Hence, the churn state is an absorbing state avoided by the optimization process of DDA.

We now define the transitions between states (represented as directional edges in Fig. 1). A player can only transit to one of two adjacent live states from current live state: 1) the player wins and advances to the first trial of the next level, i.e.,  $s_{k,t} \rightarrow s_{k+1,1}$ ; 2) loses but retries the current level, i.e.,  $s_{k,t} \rightarrow s_{k,t+1}$ . Technically, the assumption is not always true since players are able to retry the current level or play even lower levels after winning. Level replay rarely happens in most games, however. In addition to the transitions described above, all live states can directly transit to the churn state as player leave the game.

Given this structure, we need a probabilistic model of each transition that measures the likelihood of the transition happening. All outgoing transition probabilities sum to one. Since there are only three types of transitions, we can easily investigate each transition respectively.

**Level-up Transition** Starting at state  $s_{k,t}$ , players can level up to state  $s_{k+1,1}$  only if they win and do not churn. Denoting the win rate (i.e., probability to win this level at this state) as  $w_{k,t}$ , and the churn rate after winning as  $c^W_{k,t}$ , we have the level-up probability as:

$$\Pr(s_{k+1,1}|s_{k,t}) = w_{k,t}(1 - c^W_{k,t}) \quad (1)$$

**Retry Transition** From  $s_{k,t}$ , players transits to retrial state  $s_{k,t+1}$  only if they lose and do not churn. The probability of loss is  $1 - w_{k,t}$ . Denoting the churn rate after losing as  $c^L_{k,t}$ , we have the retry probability as:

$$\Pr(s_{k,t+1}|s_{k,t}) = (1 - w_{k,t})(1 - c^L_{k,t}) \quad (2)$$

**Churn Transition** Unless players make the above two transitions, they will churn. The total churn probability at  $s_{k,t}$  is the sum of the churn probabilities after winning and after losing, i.e.,

$$\Pr(churn|s_{k,t}) = w_{k,t}c^W_{k,t} + (1 - w_{k,t})c^L_{k,t} \quad (3)$$

We illustrate the transition probabilities for a given state model in Fig. 1. This probabilistic graph model is the foundation of our optimization framework for dynamic difficulty adjustment in the next section. Note that we assume  $c^W_{k,t}$  and  $c^L_{k,t}$  are independent of  $w_{k,t}$ .

## 4. ENGAGEMENT OPTIMIZATION

### 4.1 Objective Function

With the player progression model, good game design and difficulty adjustment should seek to prevent players from falling into the churn state. DDA achieves higher engagement by adjusting win rates so that the player stays on a state trajectory with lower churn rates. While existing DDA techniques adapt difficulties at each state in a greedy and heuristic manner, our framework identifies optimal win rates for all states, targeting a global objective: maximizing a player’s total engagement throughout the entire game.

Engagement indicates the amount of players’ gameplay. There are multiple engagement metrics, e.g., the number of rounds played, gameplay duration and session days. In this paper, we chose to optimize the total number of rounds played. Three reasons drive this selection. First, the number of rounds a player plays is easily measured in the progression graph. It is the transition count before reaching the churn state or completing the game. Second, many engagement metrics turn out to strongly correlate with each other. We will discuss this observation in Section 5.4. Third, maximizing the number of rounds prevents degenerate solutions that rush a player to the completion state by making the game too easy. Any solution with round repetition will score higher than the shortest path through the graph.

We use  $R$  to denote the reward function, i.e., the expected total number of rounds a player will play through the entire game. While  $R$  hardly looks tractable, we convert it to a more solvable iterative form with the help of the Markov property of the progression graph model. We define reward  $R_{k,t}$  as the expected total number of rounds played after the state  $s_{k,t}$  (level  $k$  with trial  $t$ ). Although we only consider the expectation of the reward in this paper, one could also optimize the variance.

As the player can only transit to two adjacent live states,  $s_{k+1,t}$  and  $s_{k,t+1}$ , or churn,  $R_{k,t}$  can be computed as the weighted sum of  $R_{k+1,t}$  and  $R_{k,t+1}$ . The weights are the transition probabilities between the states. Mathematically, it is written as

$$R_{k,t} = \Pr(s_{k+1,t}|s_{k,t}) \cdot R_{k+1,t} + \Pr(s_{k,t+1}|s_{k,t}) \cdot R_{k,t+1} + 1, \quad (4)$$

where  $\Pr(s_{k+1,t}|s_{k,t})$  is the probability that the player wins and levels up, and  $\Pr(s_{k,t+1}|s_{k,t})$  is the probability that one failed and retries. Adding one represents the reward by completing that round. Transition to the churn state does not contribute to the engagement.

Furthermore, substituting the transition probabilities from Eqns. 1 and 3 into Eqn. 4 (see Fig. 2), produces

$$R_{k,t} = w_{k,t}(1 - c_{k,t}^W) \cdot R_{k+1,t} + (1 - w_{k,t})(1 - c_{k,t}^L) \cdot R_{k,t+1} + 1. \quad (5)$$

Note that the original optimization objective,  $R$ , corresponding to  $R_{1,1}$ . Based on Eqn. 5,  $R_{1,1}$  is a function of win rates at all states,  $\{w_{k,t}\}$ , where  $\{c_{k,t}^W\}$  and  $\{c_{k,t}^L\}$  are parameters that can be extracted from performance data (see details in Section 4.3). Dynamic difficulty adjustment reduces to solving optimal  $\{w_{k,t}\}$  for maximizing  $R_{1,1}$ .

## 4.2 Solving for Optimal Difficulties

We need to solve an optimization problem that finds a set of optimal difficulties over all states, thus

$$\mathcal{W}^* = \arg \max_{\mathcal{W}} R_{1,1}(\mathcal{W}), \quad (6)$$

where  $\mathcal{W} = \{w_{k,t}\}$ . In practice, each  $w_{k,t}$  is constrained by game design and content. We solve for optimal  $\mathcal{W}$  under the constraint that  $w_{k,t} \in [w_{k,t}^{\text{low}}, w_{k,t}^{\text{up}}]$ .

With Eqn. 5, we can solve this optimization effectively with dynamic programming. Denoting  $R_{k,t}^*$  as the maximum reward over all possible difficulty settings, we have:

$$R_{k,t}^* = \max_{w_{k,t}} w_{k,t}(1 - c_{k,t}^W) \cdot R_{k+1,t}^* + (1 - w_{k,t})(1 - c_{k,t}^L) \cdot R_{k,t+1}^* + 1, \quad (7)$$

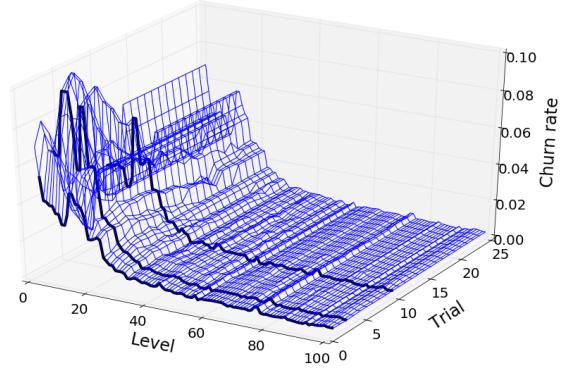
s.t.  $w_{k,t} \in [w_{k,t}^{\text{low}}, w_{k,t}^{\text{up}}]$

We can see that  $R_{k,t}^*$  is a linear function of  $w_{k,t}$  under a constraint. Therefore, the optimal win rate for state  $s_{k,t}$ ,  $w_{k,t}^*$ , can be found by:

$$w_{k,t}^* = \arg \max_{w_{k,t}} w_{k,t}(1 - c_{k,t}^W) \cdot R_{k+1,t}^* + (1 - w_{k,t})(1 - c_{k,t}^L) \cdot R_{k,t+1}^* + 1$$

$$= \arg \max_{w_{k,t}} w_{k,t}((1 - c_{k,t}^W) \cdot R_{k+1,t}^* - (1 - c_{k,t}^L) \cdot R_{k,t+1}^* + 1). \quad (8)$$

Eqn. 8 shows that given the maximal rewards of two future states,  $R_{k+1,t}^*$  and  $R_{k,t+1}^*$ , the optimal difficult  $w_{k,t}^*$  can be computed easily. As the player progression model is a directed acyclic graph, we can solve the optimization with dynamic programming. We start with a few destination states whose rewards are pre-defined and then compute the rewards of the previous states backward. The primary destination state is the end of the game,  $s_{K+1,1}$ , where  $K = k_{\max}$  is the highest level of the game. We assign zero to  $R_{K+1,1}^*$  as the reward for completion of the game. Another set of destination states are those when the number of retrials exceeds a limit, i.e.,  $s_{k,T}$  where  $T \geq t_{\max}$ . By having the upper bound of the retrial time, we can keep the progression graph to a reasonable size. This also prevents a player from too many retrials on a level when the optimization produces unrealistic results. In our experiment we set  $t_{\max} = 30$  and  $R_{k,T}^* = 1$ .



**Figure 3:** An example of a churn surface, which consists of  $c_{k,t}^L$  at all states  $s_{k,t}$ . The churn rates at 1st, 3rd, 10th trials for each level are highlighted in black to illustrate how the churn rate evolves as a player’s re-trials increases.

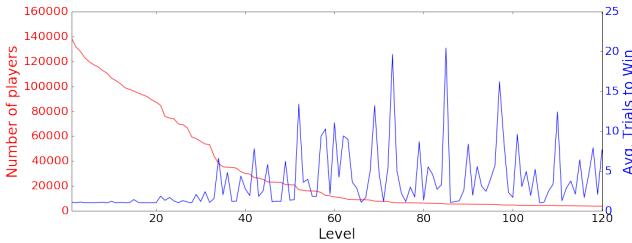
## 4.3 Churn Rates

We assume the churn rates in state transitions ( $c_{k,t}^W$  and  $c_{k,t}^L$ ) as known parameters. In practice, churn is identified as no gameplay during a period of time. We use a 7-day time frame that is common in the game industry and collect historical gameplay data of players at all states (levels and trials) to measure the churn rates. At state  $s_{k,t}$ , let  $c_{k,t}^W$  be the ratio of players who churn after winning, and  $c_{k,t}^L$  after losing. We view the churn rate over all states a two dimensions **churn surface**. Fig. 3 shows a visual example of a churn surface of  $c_{k,t}^L$ .

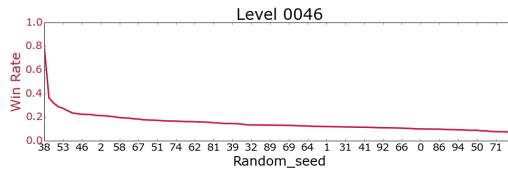
These estimates of churn rates take only two input features, level and trial, while ignoring other players’ individual features such as age, play frequency, play style, skill, performance, etc. To further improve the accuracy of churn rates, we could take advantage of long-standing churn prediction research [2, 5, 12], by employing sophisticated predictive models on individual player features to improve performance. A major challenge of using runtime churn prediction is that it increases the complexity of dynamic programming optimization. Pre-computation with various possible churn rates at each state (via bucketing) would be needed. This mechanism is not employed by our current system, but will be worth exploring in the future.

## 5. CASE STUDY: DDA IN A LIVE GAME

We now present the case study with one of our DDA implementations, a mobile match-three game developed and published by EA. Classic example of match-three genre, e.g. Candy Crush Saga by King and Bejeweled by EA, has a game board contains multiple items in different colors and shapes. A player can swap two adjacent items in each move as long as three or more items of the same color become aligned together vertically or horizontally. The aligned items will be eliminated from the board. At each level, a player needs to achieve a specific goal, for example, score a number of points in a limited number of moves. The game features more than two hundred levels. A player starts from the lowest level and advances to the higher levels. Only if a player wins the current level, the next higher level will be unlocked.



**Figure 4:** The retained population of players (red line) versus difficulties (blue line) by level. The red line represents, for each level, the number of players who have ever achieved it. The blue line represents the level difficulty, which is measured by 1/win rate, i.e., the average trials needed to win this level. We can observe the strong impact of difficulty on population retention, in particular for middle stage levels.

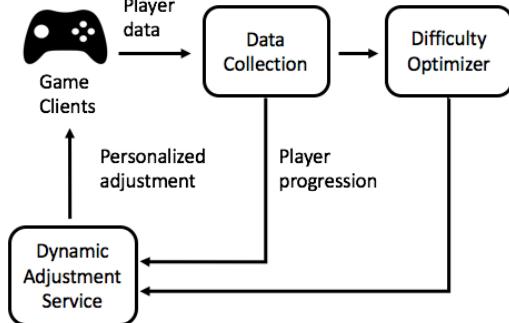


**Figure 5:** Difficulties of various random seeds at a level of the match-three game. Difficulty is measured by the win rate of a certain seed, i.e., the percentage out of all trials with this seed are actually wins. The variance of difficulties across seeds is large. We can see that the easiest seed (leftmost, seed 38) shows a win rate up to 0.75. In contrast, the hardest seed (rightmost, seed 71) has a win rate as low as 0.15.

Before adopting DDA, we must ask: can DDA help this game? To answer this question, we must convince ourselves of a causal link between the difficulty and engagement. First, we should determine if game difficulty is affecting the player engagement in the game. Second, appropriate levers should exist to effectively adjust the game difficulty. We will examine these two prerequisites in the following sections.

## 5.1 Difficulty and Engagement

To evaluate the impact of difficulty on player engagement, we compared the retained population with level difficulties (see Fig. 4). Retained population (red line) at a level is the number of players who have achieved this level as the highest one. There are players churned at each level, thus the retained population decreases as the level increases. The difficulty (blue line) is measured by the average number of trials that are needed to win this level. The more trials it takes, the more difficult this level is. Dividing all levels into three ranges: lower range ( $\leq 20$ ), middle range (21-80) and high range ( $> 80$ ), we can see that the correlation between retained population and difficulty varies. In the lower range of levels, the population naturally decays regardless of the low difficulty of these levels. Especially at level 21, there is a slight increase in difficulty, and the number of players drops significantly. In the high range of levels, the population becomes flat and decays slowly, despite that these high levels are very difficult. In contrast, in the middle range, dif-



**Figure 6:** Schematic diagram of the Dynamic Difficulty Adjustment system.

ficulty spikes are highly correlated with the steep drops in retained population. This observation supports the hypothesis that appropriate difficulty adjustment has the potential to enhance player engagement for this game.

## 5.2 Difficulty Lever

DDA adjusts win rates at different states in the progression graph through a difficulty lever. An effective difficulty lever needs to satisfy two key constraints. First, adjusting this lever should make the game easier or harder. Second, adjusting this lever should be invisible to players (as reviewed in [10]). For example, although we can simply change the “goal” or “mission” to lower game difficulty, players can easily observe it in retrials. As a consequence, such changes undermine the players’ sense of accomplishment even when they finally win with the help of DDA.

Fortunately, the case study game provides an effective difficulty lever: the random seed of board initialization. At the beginning of each round, the board is initialized from a random seed, which is indexed by a integer from 0 to 99. After evaluating the average win rate of each seed in gameplay data, we find a wide range of difficulties. Fig. 5 shows an example of difficulties versus seeds at one level. The seeds are ordered by their observed win ratios. We can see that the easiest seed (leftmost) has a win rate as high as 0.75, while the hardest seed (rightmost) has a win rate as low as 0.15. The player who plays the hardest seeds will take 5x more trials to pass this level than those playing the easiest seeds. This variance can be explained by the game mechanism. For example, some initial boards have many items of the same color close to each other, making it significantly easier to match items than boards with more pathological scattering. By carefully selecting the seed according the mapping in Fig. 5, we can control the game difficulty for players on this level. The hardest and easiest seeds provide the lower and upper bounds of win rates, i.e.,  $w^{low}$  and  $w^{up}$  in Eqn. 6.

## 5.3 Dynamic Adjustment System

We developed a real-time system to serve dynamic difficulty adjustments in an EA match-three game. Fig. 6 describes the workflow of the DDA system. At the beginning of each game round, the game clients send a request to the DDA service. The dynamic adjustment service determines the most suitable difficulty for the current player state,  $s_{k,t}$

based on the player progression and difficulty optimization results,  $w_{k,t}^*$ . The optimal win rates are computed offline as discussed in Section 4.2. Since we use random seeds as the difficulty lever, the dynamic adjustment service then applies the mapping from win rates to the random seeds showed in Fig. 5 and return it to the game client. In practice, we randomly select one seed from the top five candidate seeds to prevent from repeatedly playing only one seed. The game was first released in a period without DDA (soft launch phase), allowing the measurement of win rate for each random seed. After DDA is started, we continued collecting player data to improve the random seed mapping, churn probabilities, and difficulty optimization.

## 5.4 Experimental Results

To measure the effectiveness of our technique, we conducted A/B experiments that use multiple variants as control and treatment groups. The control group randomly recommends seeds out of all possibilities, an action corresponding to the default game behavior. The treatment group recommends the seeds based on our DDA optimization. We calculated all parameters for optimization, such as the churn surface and win rates of seeds, from real game play data. We kept track of these parameters and updated them when significant changes were observed.

The experiment started two weeks after the global release of the game. We conducted the experiment in three phases, where the proportion of the treatment group increased gradually from 10% to 40%, and finally 70% (the proportion of the control group decreases from 90%, 60% to 30%). The first two phases each lasted about one month. The third phase has been live for about four months and is ongoing. We compared core engagement metrics between the control and the treatment groups to evaluate the effectiveness of our DDA scheme. The results are daily averages normalized according to the proportion of its group, so that groups with different proportions could be compared. Phase III has not been terminated in order to collect churn probabilities and evaluate performance.

Table 1 shows the increase of the number of rounds played, suggesting that the DDA is optimizing its objective metric. In each phase of increasing treatment populations, all treatment groups exhibits statistically significant improvement ( $p\text{-value} < 0.05$ ). Table 2 shows the impact of DDA on another engagement metric, total gameplay duration. Though this metric is not the explicit optimization objective of DDA, we wanted to test an alternative hypothesis that players played more rounds in the same amount of time. Our DDA treatment group shows significant improvement on this engagement metric as well. Similarly, performance increased as more data was collected in the second phase; then stayed stationary when the gameplay data became accurate and stable in the latest phase. Note that we see slightly different performances between iOS and Android platforms, though, in the same order. This resonates with the common observation in mobile game development that user behaviors between platforms often differ from each other [1, 14], so that separate modeling and treatment are preferred.

We observed the lowest performance in Phase I, when DDA is completely based on the model we learned from limited data - soft launch data and first two weeks in worldwide release. The soft launch data is only partially useful as some game design is changed before worldwide release. After

| Phase | Platform | Default   | DDA       | Delta   | Gain  |
|-------|----------|-----------|-----------|---------|-------|
| I     | iOS      | 1,118,237 | 1,167,616 | +49,379 | +4.4% |
|       | Android  | 789,640   | 825,182   | +35,543 | +4.5% |
| II    | iOS      | 855,267   | 915,334   | +60,067 | +7.0% |
|       | Android  | 1,137,479 | 1,228,473 | +90,995 | +7.9% |
| III   | iOS      | 711,749   | 763,508   | +51,759 | +7.2% |
|       | Android  | 1,285,448 | 1,366,820 | +81,373 | +6.3% |

**Table 1:** Total numbers of rounds played daily in the default (control) group versus in the DDA treated group. Delta is the absolute increase by DDA and Gain is the relative increase.

| Phase | Platform | Default   | DDA       | Delta    | Gain  |
|-------|----------|-----------|-----------|----------|-------|
| I     | iOS      | 3,684,082 | 3,847,516 | +163,435 | +4.4% |
|       | Android  | 2,686,781 | 2,814,953 | +128,172 | +4.8% |
| II    | iOS      | 2,916,570 | 3,148,722 | +232,152 | +7.9% |
|       | Android  | 3,787,414 | 4,129,762 | +342,348 | +9.0% |
| III   | iOS      | 2,582,809 | 2,788,690 | +205,881 | +8.0% |
|       | Android  | 4,619,907 | 4,956,680 | +336,773 | +7.3% |

**Table 2:** Total durations of gameplay time (in minutes) daily in the control group versus in the DDA treated group.

the release, we continued to collect more data to form more accurate parameters for DDA optimization. This explains the further improved performance in Phase II over Phase I. In Phase III, the performance has been observed stable for more than three months. The amount of boost is relatively smaller than that of Phase II because there are fewer new players in this phase and DDA has higher impacts on early stage players (see Fig. 4).

Last but not least, we also compared the impact on monetization between the control and the treatment groups. This comparison is critical as a monetization objective might contradict engagement maximization. Our DDA treatment group had a neutral impact on monetization. No statistically significant difference on in-game transaction revenues was observed between two groups. This is probably caused by the mechanism of our optimization framework: it “saves” players of high churn risks, who are less likely to spend.

## 6. CONCLUSION

In this paper, we presented a framework that approaches dynamic difficulty adjustment (DDA) as an optimization problem. While existing DDA systems adapt game difficulty in a greedy manner for local benefit, our method maximizes the player engagement throughout the entire game. We modeled the player progression as a probabilistic graph, with which engagement maximization becomes a well-defined objective and we proposed an efficient dynamic programming method to solve it. We evaluated an implementation of the DDA technique in a mobile game by EA. The DDA treated group shows significant increases in core engagement metrics, such as a total number of rounds played and gameplay duration, while it is revenue neutral compared to the control group with no DDA enabled.

In the near future, we will extend the framework to a wide range of game genres. The key to successfully applying DDA to other genres is the construction of an adequate progression model. For level-based games, the states can be naturally identified by two major dimensions: level and trial. For more complicated games with non-linear or multi-

ple progressions, such as role-playing games (RPG), we can also define the states with different dimensions. The graph might be more complicated as it includes more states and connections. Solving the optimal difficulty associated with each state, which maximizes player engagement, however, remains a well-defined graph optimization problem.

Another problem worth investigating is the cold-start stage when the graph parameters (such as seed difficulty and churn risks) needed for optimization are not known yet. In the current system, we learned these parameters in soft launch and the first few weeks after release before starting treatment (see Section 5.4). A possible direction is to conduct reinforcement learning at this early state, by defining some short-term awards that help quickly establish greedy policies. Once sufficient data is collected, we can then shift to the supervised optimization framework.

Last but not least, we would like to explore generic dynamic player experience. Beyond difficulty adjustment, how can we provide a dynamic and intelligent game experience personalized for individual players? It includes, but not limited to, dynamic tutorials, dynamic messages, dynamic awards, dynamic advertisement and etc. We believe that the progression model and corresponding optimization framework will become key to these generic solutions.

## 7. REFERENCES

- [1] Z. Benenson, F. Gassmann, and L. Reinfelder. Android and ios users' differences concerning security and privacy. In *CHI '13 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '13, pages 817–822, New York, NY, USA, 2013. ACM.
- [2] R. Bernhaupt. User experience evaluation in entertainment. In *Evaluating User Experience in Games*, pages 3–7. Springer, 2010.
- [3] M. Csikszentmihalyi and I. S. Csikszentmihalyi. *Optimal experience: Psychological studies of flow in consciousness*. Cambridge university press, 1992.
- [4] P. Demasi and J. d. O. Adriano. On-line coevolution for action games. *International Journal of Intelligent Games & Simulation*, 2(2), 2003.
- [5] F. Hadjii, R. Sifa, A. Drachen, C. Thurau, K. Kersting, and C. Bauckhage. Predicting player churn in the wild. In *2014 IEEE Conference on Computational Intelligence and Games*, pages 1–8. IEEE, 2014.
- [6] J. Hagelback and S. J. Johansson. Measuring player experience on runtime dynamic difficulty scaling in an rts game. In *2009 IEEE Symposium on Computational Intelligence and Games*, pages 46–52. IEEE, 2009.
- [7] G. Hawkins, K. Nesbitt, and S. Brown. Dynamic difficulty balancing for cautious players and risk takers. *Int. J. Comput. Games Technol.*, 2012:3:3–3:3, Jan. 2012.
- [8] R. Hunicke. The case for dynamic difficulty adjustment in games. In *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology*, pages 429–433. ACM, 2005.
- [9] M. Jennings-Teats, G. Smith, and N. Wardrip-Fruin. Polymorph: dynamic difficulty adjustment through level generation. In *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*, page 11. ACM, 2010.
- [10] S. Karpinskyj, F. Zambetta, and L. Cavedon. Video game personalisation techniques: A comprehensive survey. *Entertainment Computing*, 5(4):211–218, 2014.
- [11] O. Missura and T. Gärtner. Predicting dynamic difficulty. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 2007–2015. Curran Associates, Inc., 2011.
- [12] J. Runge, P. Gao, F. Garcin, and B. Faltings. Churn prediction for high-value players in casual social games. In *2014 IEEE Conference on Computational Intelligence and Games*, pages 1–8. IEEE, 2014.
- [13] P. Spronck, I. Sprinkhuizen-Kuyper, and E. Postma. Difficulty scaling of game AI. In *Proceedings of the 5th International Conference on Intelligent Games and Simulation (GAME-ON 2004)*, pages 33–37, 2004.
- [14] C. Zhou, Y. Guo, Y. Chen, X. Nie, and W. Zhu. Characterizing user watching behavior and video quality in mobile devices. In *2014 23rd International Conference on Computer Communication and Networks (ICCCN)*, pages 1–6, Aug 2014.
- [15] A. Zook, S. Lee-Urban, M. R. Drinkwater, and M. O. Riedl. Skill-based mission generation: A data-driven temporal player modeling approach. In *Proceedings of the The Third Workshop on Procedural Content Generation in Games*, PCG'12, pages 6:1–6:8, New York, NY, USA, 2012. ACM.
- [16] A. Zook and M. O. Riedl. A temporal data-driven player model for dynamic difficulty adjustment. In *Artificial Intelligence and Interactive Digital Entertainment*, 2012.

# Enemy Within: Long-term Motivation Effects of Deep Player Behavior Models for Dynamic Difficulty Adjustment

Johannes Pfau

Digital Media Lab, TZI,  
University of Bremen  
Bremen, Germany  
jpfau@tzi.de

Jan David Smeddinck

Open Lab, School of Comp.,  
Newcastle University  
Newcastle upon Tyne, UK  
jan.smeddinck@newcastle.ac.uk

Rainer Malaka

Digital Media Lab, TZI,  
University of Bremen  
Bremen, Germany  
malaka@tzi.de

## ABSTRACT

Balancing games and producing content that remains interesting and challenging is a major cost factor in the design and maintenance of games. Dynamic difficulty adjustment (DDA) can successfully tune challenge levels to player abilities, but when implemented with classic heuristic parameter tuning (HPT) often turns out to be very noticeable, e.g. as “rubberbanding”. Deep learning techniques can be employed for deep player behavior modeling (DPBM), enabling more complex adaptivity, but effects over frequent and longer-lasting game engagements, as well as comparisons to HPT have not been empirically investigated. We present a situated study of the effects of DDA via DPBM as compared to HPT on intrinsic motivation, perceived challenge and player motivation in a real-world MMORPG. The results indicate that DPBM can lead to significant improvements in intrinsic motivation and players prefer game experience episodes featuring DPBM over experience episodes with classic difficulty management.

## CCS Concepts

- Human-centered computing → User models;
- Computing methodologies → Neural networks;
- Applied computing → Computer games;

## Author Keywords

Dynamic difficulty adjustment; Player Modeling; Neural Networks; Deep Learning; MMORPGs; Games

## INTRODUCTION

With the ongoing rise of complexity, popularity and content production cost of video game development, consistently balanced challenges that keep players motivated over the long term are becoming hard to attain, especially with large player communities encompassing broad ranges of proficiency. Dynamic Difficulty Adjustment (DDA) [23] denotes the principle of adapting video game challenges to players' abilities - both mental and physical / dexterity - in order to allow motivation-fostering flow states [10] to arise. It has been successfully

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

CHI '20, April 25–30, 2020, Honolulu, HI, USA.

© 2020 Copyright is held by the owner/author(s). Publication rights licensed to ACM. ACM ISBN 978-1-4503-6708-0/20/04 ..\$15.00.  
http://dx.doi.org/10.1145/3313831.3376423

deployed in scientific [23, 24, 46] and industrial [7, 16, 55] contexts and is usually accomplished by continuous tuning of core game variables (such as speed, damage or hit ratio). However, these systems are inherently limited to a small number of high-level parameters, require careful tuning of thresholds in heuristic parameter tuning (HPT) [51] and have to be hidden to avoid exploitation; e.g. as not to incentivize players to perform badly on purpose [43]. This results in limited expressivity and complexity of system behavior, as well as in considerable development cost. To address these limitations of classic HPT, we present a novel DDA strategy by implicitly assessing individual player proficiency using Deep Player Behavior Modeling (DPBM) [38] and generating adaptive, personalized challenges. Player behavior, in terms of state-action decision making, is captured while fighting an in-game opponent and trained onto an individual, initially randomized model. Upon the next encounter, the opponent uses this model generatively by retrieving action probabilities for each game state emerging in an interaction. As a consequence, its decision making approximates the original player's behavior, implicitly representing the particular game proficiency. Evaluating the real-world applicability of DPBM, we aim to answer the following research questions:

- Do players perceive behavior from DPBM as representative of their own decision making?
- Is the players' self-reported intrinsic motivation when interacting with a DPBM opponent higher than for traditional HPT encounters?
- Can we measure a substantial long-term motivation achieved by DPBM?

We hypothesize that an agent that keeps up with the progress of the player, displays similar weaknesses and challenges players to constantly improve or rethink strategies will yield a novel and captivating take on DDA. For the purpose of evaluating the differences between HPT and DPBM, we implemented *Eternal Challenge* – an adaptive instance dungeon inside of the popular Massively Multiplayer Online Role-Playing Game (MMORPG) *Aion* [34] – and assessed players' motivation in a field study ( $n = 171$ ) on an existing private server. After a deployment of four weeks, we were successful in showing a significantly higher long-term usage of the instance compared to all alternatives and that the DPBM opponent contributes significantly more to this motivation than HPT mechanics,



Figure 1. Appearance of the different opponents with DDA through HPT variables and DPBM.

based on quantitative – and supported by qualitative – insights. Furthermore, some players stated that they noted the progress of the DPBM opponent in learning their own individual strategies, leading to a unique game experience. We contribute to games user research and inform game development by investigating DPBM as a form of novel DDA in a situated medium- to long-term study, showcasing its distinct potential for fostering intrinsic motivation and demonstrating a working approach with learning adaptive opponents in the wild.

## RELATED WORK

Providing and balancing an accurate level of difficulty is critical for keeping players constantly engaged [2, 23]. Disparities can ultimately lead to **boredom/underload or frustration/overload**, which make for two of the main causes why players stop playing games [11]. Since individual skill and its progression are hard to foresee throughout potentially large player bases and difficulty and it's progression can not be defined or programmed precisely, the field of DDA attempts to regulate emergent mismatches dynamically. To estimate imbalanced challenge-proficiency-discrepancies, various assessment techniques have been researched, such as success probability estimation [24, 32] or biofeedback [22, 25, 35]. However, When it comes to adjusting this difficulty, most approaches focus on HPT (apart from procedural world generation [26, 57, 50]), even in the most recent advancements [3, 4, 8, 17, 18, 40].

Opponents that imitate the player character exist in numerous commercial games, perhaps most notably the recurring *Dark Link* in the *The Legend of Zelda* series [15], the *Guild Wars Doppelganger* [5], *Renegade Shepard* from *Mass Effect 3* [6] and *SA-X* in *Metroid Fusion* [42]. Yet, so far these have only been realized as crude approximations of the original player, as they mimic appearance, equipment, basic moves and/or skill sets by relying on heuristic, strategically rigid decision-making.

At the same time, machine learning approaches in video games that harness continuous improvement through simulated play have become popular, e.g. the deep reinforcement learning of *Atari* games [31], temporal difference learning of *Backgammon* [54] or the surpassing of human player performance in the board game that has been rated as not solvable by artificial intelligence methods for a long time; *Go* [48]. The field of player modeling has seen explorations of machine learning techniques for multiple purposes, prominently featuring

prediction, classification or analysis [13, 14, 27, 53], to facilitate individual game interpretations, testing or for providing believable opponents. Still, the incorporation of machine learning approaches for generative player modeling for DDA and the resulting player experiences remain under-investigated. Holmgård et al. studied *personas* for player decision modeling [19, 20] that continually observe and adapt to human behavior in order to produce agents with different decision-making styles. These *personas* were realized via evolutionary linear perceptrons and compared to heuristic agents in a test bed 2D dungeon crawler game, resulting in a higher player-rated human-likeness. They also assessed player models when defined as “deviations from theoretically rational actions” in a study of *Super Mario Bros.* [1, 21] and clustered these by means of feature extraction. Using the same game, Ortega et al. [36] imitated human playing styles by means of Neuroevolution and Dynamic Scripting and reached higher scores of human-likeness than performance-directed AI agents, based on subjective judgments. Missura and Gärtner utilized Player Modeling in a 2D test bed shooter via Support Vector Machines acting as predictors for difficulty mismatches and enabling classical DDA parameter tuning based on the results [30]. In previous work, we were successful in showing that player model agents can yield significantly higher motivation compared with heuristic opponents in a short-term online study using the 2D platform fighter *Korona:Nemesis* [9, 39]. Based on these insights, player awareness about substituting individual players with DPBM agents in online multi-player matches was also assessed. In contrast to heuristic bots, DPBM agents turned out to be indistinguishable from their human precursors [37]. In addition, we contrasted different machine learning techniques in a player modeling study of the MMORPG *Lineage 2* [33, 38]. Deep learning offered the best individual prediction accuracy, facilitating the production of playing sessions that closely resemble the original behavior, as well as for differentiating between players. Consequently, we discussed the broader implications for the application of DPBM in: DDA (offering adaptation beyond parameter tuning; training players by exposing them to own strengths and weaknesses), player substitution (bridging online match disruption due to dropouts; providing more individually representative agents), automated game testing (enhancing the estimation of balancing issues by incorporating realistic human player behavior) and cheating detection (revealing behavior that is more likely to stem from undesirable third-party bots rather than players; yield-

ing objective evidence based on behavior in cases of identity theft).

To the best of our knowledge, there is no prior work assessing the experience of players who continuously challenge themselves, where generative player modeling facilitates proficiency progress.

## APPROACH

In contrast to the aforementioned studies, the approach presented in this work provides a medium- to long-term situated evaluation. We compare the deployment of player modeling through DPBM with traditional HPT and assess feasibility, approval and motivation in a complex AAA game through a highly ecologically valid field study. To facilitate realistic and generalizable results that avoid artificial laboratory study setting biases [44], we aimed for the deployment of our approach in a real-world setting in a fully fledged game with an existing community of players. In the following, we explain the construction of the recorded training data format, how it was informed by expert interviews, the DPBM architecture, and the study environment.

### Expert Interview

In order to gain a more elaborate understanding of viable strategies, decision making and what behavior might lead to different play styles in *Aion*, one of the authors consulted 3 expert players of the game (each with 7–8 years of prior experience) and extracted the most important aspects qualitatively, using brief 1 hour semi-structured interviews over the course of one day. Apart from a less-structured introduction and follow-up discussion after each item, we asked the following questions:

- In a one-on-one situation against a (computer controlled opponent / human player), based on which factors do you decide which skill to use?
- How do you react when you are not able to execute your strategy?
- How would you approach an opponent of the same class that is equally proficient as yourself?

We analyzed the interview using an outcome-oriented structuring content analysis after Mayring [28] and consolidated the most expressive statements about factors that qualify as good indicators for decision making. The most descriptive factors as indicated by the experts are adherence to skill *rotations* and situational *responsive decisions*. Within *rotations*, expert players predominantly apply a specific set of preferred sequences of actions, e.g. ramping up damage by combinations of enhancing and weakening skills or controlling the opponent by a succession of restricting skills. Due to the large amount of possible skills or items to use (cf. Figure 2), these rotations can include complex chains of consecutive skills and/or contain *sub-rotations*. *Responsive decisions* denote the reaction to certain states that the player character or an opponent is in, e.g. healing oneself when hit points are low, removing restricting conditions on the character, increasing or reducing distance between characters or exploiting temporary

conditions the opponent is in. They can also trigger more complex situation-specific *rotations*. The description of play-style aspects by means of *rotations* and *responsive decisions* is not limited to this specific game, but broadly generalizes to the genre of action RPG games. We defined the DPBM state-action architecture on the basis of these factors, including player and target state information as crucial indicators for *responsive decisions* and previous skill information for positioning within *rotations*.

In combination, these factors compose the game state that is fed into the DPBM input layer, while the output layer is trained according to the respective skill that the player used in this situation (cf. Figure 3).

### Adaptive Instance Dungeon

Instance dungeons are a major part of MMORPGs, as they can be entered numerous times, solo or in a group, to acquire experience points, equipment, currencies and/or other desirable items. As such, they provide a fertile testing ground for evaluating long-term motivation [52], since most often repeated or even continuous entries are required to reach higher-level goals. To gather expressive evidence of the motivational potential of DPBM, we developed the single-player adaptive instance dungeon *Eternal Challenge* that incorporates both traditional DDA aspects via HPT as well as DPBM. Within the instance, the players encountered various opponents that were clearly distinguishable by their visual appearance (cf. Figure 1) and were adjusted through distinct parameters tuned by HPT (cf. Table 1). The underlying proficiency variable  $\lambda$  approximated the player's performance level by being increased whenever he successfully completed *Eternal Challenge* and decreased at the characters death or temporal expiry of the countdown. This way, HPT produces a typical “rubberbanding” effect between player-specific thresholds of lack of challenge and excessive challenge, which is one of the most common ways to enable flow-states to arise in traditional DDA [47] and was constructed by following the inspiration of these approaches [17, 23, 30, 47] in combination with fine-tuning by the developers operating the server to find a range covering too easy, too hard and enough configurability in between for every observed player.



**Figure 2.** Exemplary arrangement of a subset of skills available to the Sorcerer class in *Aion*. Additionally, context-dependent skills (when the player or a target opponent is in a particular condition) and a multitude of items can be activated.

|                            |  |
|----------------------------|--|
| $\lambda$ difficulty level | Increased whenever <i>Eternal Challenge</i> was completed successfully, decreased upon death or timeout.   |
| $\alpha$ frequency         | With increased $\lambda$ , the temporal spawn delay of $\alpha$ opponents was decreased, resulting in an exponential increase of difficulty.                         |
| $\beta$ perseverance       | With increased $\lambda$ , hit points (HP) of $\beta$ opponents increased, making them harder / more time-consuming to defeat.                                       |
| $\gamma$ disturbance       | With increased $\lambda$ , $\gamma$ opponents used more actions that weaken the player, which decreases survivability, damage performance and increases the tension. |
| DPBM                       | No explicit parameter tuning was used, since network proficiency approximates the player's skill implicitly.   |

Table 1. DDA mechanisms of *Eternal Challenge*, mapping  $\lambda$  to the difficulty of  $\alpha, \beta, \gamma$ , while DPBM seeks to emulate the player's behavior.

To avoid incentivizing players to perform badly on purpose, rewards (in the form of experience gained and the level-range of items dropped) were adjusted to be proportional to the difficulty level  $\lambda$ . Upon entering the instance, a 15-minute countdown started that expelled the player if it was not finished after expiration. Within this time limit, the player was expected to destroy a sturdy, non-responsive opponent ( $\beta$ ) that spawned additional, hostile enemies over time ( $\alpha, \beta, \gamma$ ) which had to be endured or defeated as well. As soon as the main opponent was defeated, an additional foe that utilizes DPBM appeared in an adjacent room. If the player managed to beat this opponent as well, rewards were distributed and the internal  $\lambda$  level was raised accordingly.  $\lambda$  had no theoretical, but a practical upper limit, since the game inherently restricts reaching damage per second values beyond a certain threshold.

### Deep Player Behavior Modeling

When entering *Eternal challenge*, the recorded behavioral data from all preceding runs of the player was retrieved from the underlying database and fed into a feed-forward multi-layer perceptron with backpropagation and a logistic sigmoid activation function (cf. Figure 3), where input and output layer size varied depending on the player's class, skill set and usage ( $M = 98.2, SD = 15.1$ ) input,  $5 \times 10$  hidden, ( $M = 76.2, SD = 15.1$ ) output nodes). The network was initialized randomly and trained over 1000 epochs, based on the insights of previous work [38, 39] and benchmarks prior to the study that indicated diminishing returns when further increasing the range of parameters. When encountering the DPBM opponent, the trained model was applied generatively to retrieve a set of action probabilities given the occurring state description at real-time. After a weighted choice, the resulting skill was executed, followed by querying the DPBM

for the next situation, effectively approximating the learned behavior from the player's preceding battles. As movement was controlled heuristically, temporal-dynamics of behavior are not explicitly modeled, but behavior over time is modeled by focusing the sequencing of skill *rotations* and *responsive decisions* in each occurring state. In terms of the player modeling taxonomy of Yannakakis et al. [58], this implementation realizes a *model-free* (bottom-up) player modeling approach mapping *gameplay data* to actions via *preference learning* and *classification*. According to the player modeling description framework of Smith et al. [49], DPBM directly utilizes *game actions* (**domain**) to generate (**purpose**) individually (**scope**) modeled behavior by means of induced (**source**) training of machine learning techniques.

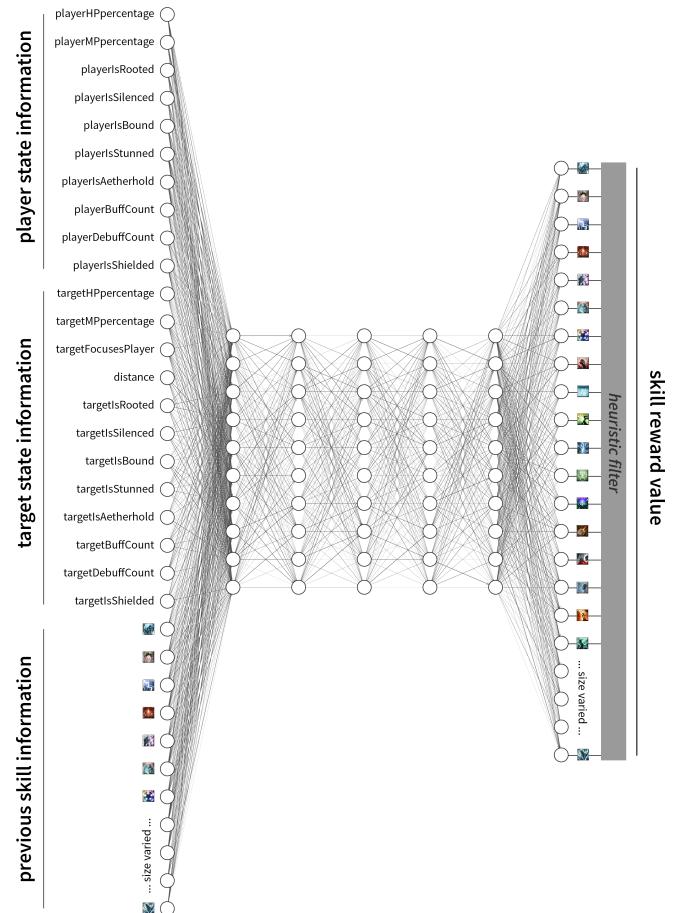


Figure 3. The DPBM architecture mapping game state (information about player, opponent and skill history) to action (skill usage) probabilities. The size of the input and output layers varied depending on the player's class, skill set and usage. The resulting action probability array is filtered heuristically by removing skills that are impossible to execute due to cool-down, MP shortage or other insufficient conditions.

## STUDY

In order to evaluate the DDA and intrinsic motivation capabilities of DPBM, we conducted an online field study following a within-subjects design over the course of four weeks. The adaptive instance was published on a private *Aion* game server. To ensure that the measured motivation is attributable to the DPBM approach, we took several precautions. In order to minimize novelty or anticipation bias, we did not announce the existence or concept of the instance prior release and chose a long-term study design. In addition, rewards constitute one of the biggest extrinsic motivators for long-term commitments [56] and thus a potentially high confounding factor when assessing intrinsic motivation. Therefore, the rewards of *Eternal Challenge* were kept conservative and approximated the amount of rewards in other available instances, i.e. players were not able to obtain something that they would not be able to elsewhere and did not acquire a higher reward-to-time-ratio. Findings of Deci et al. [12] also suggest that excessive extrinsic rewards can inhibit intrinsic motivation, the core factor of engagement and enjoyment [45, 41]. Finally, as interplay among players is a major motivating factor of MMORPGs [56], we excluded multi-player situations, leaderboards, high score lists or the publication of ranks and completion times during the study period to avoid complex potential biases in this early situated study. Although MMORPGs are designed to be about multi-player scenarios, occasions of playing solo occur on a regular basis and novel challenge paradigms should arguably be tested in basic, controllable setups before being extended to include additional factors, such as team play or competition.

## Measures

For every single *Eternal Challenge* run performed by a player, we recorded state-action data for DPBM training (cf. Figure 3), instance completion times and results (failed or succeeded), as well as the training times and prediction accuracies of the DPBM. In addition, we logged entry counts and timestamps for all available instance dungeons for further activity comparisons. After the study period, a post-study questionnaire asked for player-reported assessments of *perceived competence*, *interest-enjoyment*, *tension-pressure* and *effort-importance*, following the *Intrinsic Motivation Inventory* (IMI) [29], for comparison between the traditional DDA parameters  $\alpha$ ,  $\beta$ ,  $\gamma$ , and the DPBM opponent. Each iteration of the questionnaire was explicitly headed by a display of the appearance of the corresponding opponent in order to assure correctly targeted responses (cf. Figure 1).

Additionally, the survey contained qualitative queries about the appreciation of – and strategies used against – the opponents, the impression of DPBM opponent’s behavior in the players’ own words, and a free field for additional remarks.

## Procedure

The instance dungeon *Eternal Challenge* was introduced and released as part of a regular update to the private server. From then on, players of the community had the opportunity to enter it up to once daily, independent from entering different or additional instances. After four weeks, the recording of in-game data stopped and the post-study questionnaire was advertised on a message board associated with the server.

## Participants

During the study period, ( $n = 171$ ) participants entered *Eternal Challenge* resulting in 776 total instance runs. 30 players (17 men, 13 women (self-identified)) completed the optional post-study questionnaire.

## RESULTS

Using a one-way RM ANOVA, we found significant effects for the IMI scores *perceived competence*, *interest-enjoyment*, *tension-pressure* and *effort-importance* between conditions  $\alpha$ ,  $\beta$ ,  $\gamma$  and DPBM (cf. Table 2).

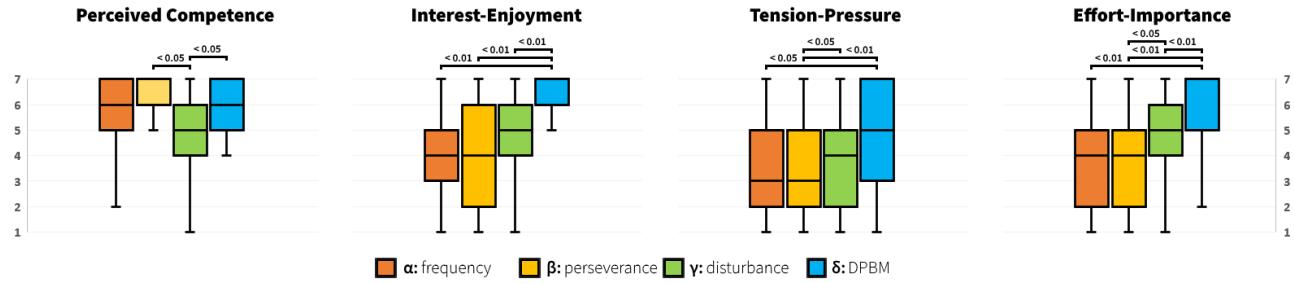
These outcomes were further evaluated using two-tailed paired t-tests (cf. Table 3, Figure 4). Employing conservative Bonferroni correction,  $p$ -values were multiplied with the amount of repeated comparisons. DPBM received significantly higher scores of *interest-enjoyment* and *effort-importance* compared to all HPT opponents, resulting in mostly large effect sizes after Cohen. It also outperformed  $\alpha$  and  $\beta$  significantly in terms of *tension-pressure* with medium effect sizes.

|                             |                   |           |                 |
|-----------------------------|-------------------|-----------|-----------------|
| <i>perceived competence</i> | $F(3, 26) = 3.59$ | $p < .05$ | $\eta^2 = 0.29$ |
| <i>interest-enjoyment</i>   | $F(3, 26) = 8.75$ | $p < .01$ | $\eta^2 = 0.5$  |
| <i>tension-pressure</i>     | $F(3, 26) = 3.37$ | $p < .05$ | $\eta^2 = 0.28$ |
| <i>effort-importance</i>    | $F(3, 26) = 5.63$ | $p < .01$ | $\eta^2 = 0.39$ |

**Table 2.** ANOVA results ( $F(df_1, df_2)$ –,  $p$ –values,  $\eta^2$  for effect size) of the *Intrinsic Motivation Inventory* between the different opponents.

On average, players spent ( $M = 7.71, SD = 2.49$ ) minutes in the instance and used up to 91 ( $M = 21.1, SD = 11.6$ ) different skills against the DPBM opponent. Model training times lasted ( $M = 2018, SD = 3692$ ) ms per session with ( $M = 8.75, SD = 3.34$ ) ms per recorded skill.

To assess the objective quality of the underlying machine learning model and render it comparable to related approaches, 80% of the data recorded until any given time of entry into the instance was used for training, whereas 20% served for a routine initial test, resulting in ( $M = 60.64, SD = 22.57$ )% prediction accuracy.



**Figure 4.** Intrinsic motivation inventory (IMI) results for the compared DDA variables. Includes medians (center marks), standard deviations (boxes), minimal and maximal values (whiskers) and significant difference markers.

Compared to all 35 available instances in the game, *Eternal Challenge* (EC) became the most popular instance by daily numbers of players over the duration of the study (cf. Figure 5), as chi-square goodness of fit tests show (cf. Table 4), assuming equal proportions. Even when omitting the first quarter to counteract a presumable novelty bias in the distinct initial spike, EC still outmatched all alternatives.

|                    | DPBM   |                      | DPBM   |
|--------------------|--|----------------------|--|
| interest-enjoyment | ( <i>M</i> = 6.17,<br><i>SD</i> = 1.11)                    | effort-importance    | ( <i>M</i> = 5.87,<br><i>SD</i> = 1.63)                    |
| α                  | <i>p</i> = .000<br>( <i>M</i> = 4,<br><i>SD</i> = 1.51)    | α                    | <i>p</i> = .006<br>( <i>M</i> = 4.04,<br><i>SD</i> = 2.03) |
| β                  | <i>p</i> = .001<br>( <i>M</i> = 4.04,<br><i>SD</i> = 2.06) | β                    | <i>p</i> = .000<br>( <i>M</i> = 3.91,<br><i>SD</i> = 1.83) |
| γ                  | <i>p</i> = .01<br>( <i>M</i> = 4.78,<br><i>SD</i> = 1.76)  | γ                    | <i>p</i> = .004<br>( <i>M</i> = 4.91,<br><i>SD</i> = 1.81) |
| tension-pressure   | DPBM<br>( <i>M</i> = 4.91,<br><i>SD</i> = 2.15)            | perceived competence | DPBM<br>( <i>M</i> = 5.83,<br><i>SD</i> = 1.07)            |
| α                  | <i>p</i> = .049<br>( <i>M</i> = 3.39,<br><i>SD</i> = 1.8)  | α                    | <i>p</i> > .05<br>( <i>M</i> = 5.65,<br><i>SD</i> = 1.67)  |
| β                  | <i>p</i> = .007<br>( <i>M</i> = 3.39,<br><i>SD</i> = 1.8)  | β                    | <i>p</i> > .05<br>( <i>M</i> = 5.91,<br><i>SD</i> = 1.35)  |
| γ                  | <i>p</i> > .05<br>( <i>M</i> = 4.17,<br><i>SD</i> = 1.85)  | γ                    | <i>p</i> > .05<br>( <i>M</i> = 4.65,<br><i>SD</i> = 1.72)  |

**Table 3.** Means, standard deviations and significant t-test results after Bonferroni correction (*p*-values and Cohen's *d* for effect size, *df* = 29) of the Intrinsic Motivation Inventory between the different opponents.

For the qualitative remarks, we used a structuring content analysis after Mayring [28] to assess the effect of challenging the DPBM opponent. Players were asked to state their general impression and opinion freely, without confounding or influencing questions. From the utilizable statements, 31.8% describe an appropriate challenge (e.g. “quite easy at first but afterwards I really was busy thinking about how I approach him”, “it’s almost as good as I am”), while 9.1% depict it as slightly too high or slightly to low. 13.6% emphasize a notable entertaining factor, whereas 4.4% declare that this encounter did not appeal to them. Although the behavior or decision-making of the DPBM opponent was never explicitly stated or explained during the study, 36.4% of players ascribed the ability to learn from previous battles and the adaptation to the player’s own behavior, combos, rotations and/or strategies to their enemy (e.g. “at first he randomly used skills that I also used, later he added my combos”, “tried to replicate my own skills and techniques”, “it was hilarious when I played against myself”).

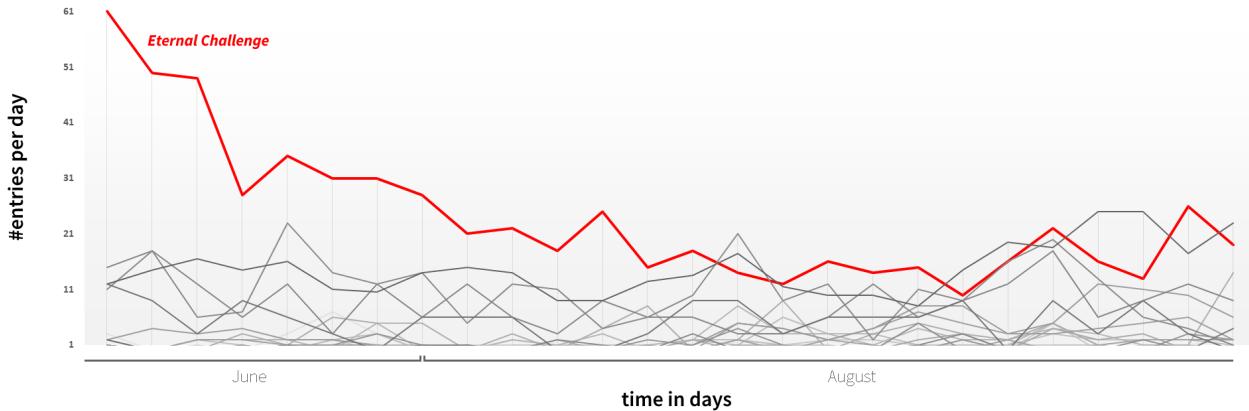
| During complete study period: |                               |                 |  |
|-------------------------------|-------------------------------|-----------------|--|
| EC vs. #2                     | $\chi^2(1, n = 1007) = 58.64$ | <i>p</i> < 0.01 |  |
| EC vs. #3                     | $\chi^2(1, n = 902) = 134.26$ | <i>p</i> < 0.01 |  |
| EC vs. #4                     | $\chi^2(1, n = 856) = 181.35$ | <i>p</i> < 0.01 |  |

| After first quarter of study period: |                              |                 |  |
|--------------------------------------|------------------------------|-----------------|--|
| EC vs. #2                            | $\chi^2(1, n = 627) = 4.48$  | <i>p</i> = 0.03 |  |
| EC vs. #3                            | $\chi^2(1, n = 526) = 45.09$ | <i>p</i> < 0.01 |  |
| EC vs. #4                            | $\chi^2(1, n = 493) = 70.93$ | <i>p</i> < 0.01 |  |

**Table 4.** Chi-square goodness of fit tests between *Eternal Challenge* and the second, third and fourth most popular instance. Less popular instances have shown similarly significant results, but have been omitted for the sake of readability.

## DISCUSSION

Our results indicate distinct effects on approval and intrinsic motivation for DPBM opponents, as well as effects on long-term commitment for the presence of DDA in general. We were successful in evidencing significantly higher motivation for players to enter adaptive instance dungeons compared to static alternatives over considerable duration of successive play sessions and report indications that DPBM attributes significantly more to this preference than traditional DDA parameter tuning.



**Figure 5.** Daily number of unique players entering *Eternal Challenge* compared to all other available instances during the study period.

This insight is based on notably high absolute IMI scores and significant differences compared to conventional DDA techniques. The outcome that DPBM outperformed HPT in terms of *interest-enjoyment* indicates a high “fun factor”, while *tension-pressure* and *effort-importance* highlight the considerable challenge, leading to an overall higher intrinsic motivation and linked potential to induce flow. The actual implicit DDA capabilities of DPBM are backed by qualitative statements that reveal an appropriate challenge, a noticeable difficulty adjustment over time and the perception of playing against an equal opponent that facilitates rethinking of habitual behavior. This work also demonstrates the technical applicability of large-scale, long-term generative player modeling with reasonable training times and accuracies.

Overall, our work provides quantitative and qualitative empirical evidence supporting our initial hypotheses about facilitating long-term motivation potential, capabilities for enabling DDA and individual representation, indicating the following responses to the respective research questions:

- We measured a substantially and consistent motivation achieved with the support of DPBM over the medium-to long-term.
- The measured intrinsic motivation of challenging a DPBM-fuelled opponent significantly exceeded traditional HPT.
- Players perceive behavior from DPBM as representative of – or comparable to – their own decision-making.

#### Limitations and Future Work

The mixed-bag fashion of the instance, which resulted from aiming to maintain an ecologically-valid realistic instance design, results in a combined experience of HPT and DPBM opponents that might influence the participants’ assertions. This study design was selected due to the long-term period of the study in a community where players know each other, rendering a between-subjects design confounding, since players would have exchanged views about the different conditions and/or complained about unjust treatments.

To further corroborate evidence and to gain a clear comparison between the different HPT factors and DPBM, the experiment should be replicated to manifest a control group (mutually exclusive from this player base) in which no DPBM (or HPT) is present. Apart from that, the Intrinsic Motivation Inventory was designed to measure single sessions within experiments. While it was not explicitly developed for this study’s setup, we found it to be the most appropriate questionnaire to assess motivation, as there is no validated reflective long-term motivation questionnaire that does not have to be raised after every single session (which was omitted in favor of ecological validity).

Based on our achievements and outcomes, we are looking forward to extending the scope of using DPBM in video games to enabling personalized, adaptive challenges that go beyond one-on-one situations to encompass interactions between different players and consider both competitive as well as cooperative interplay. DPBM agents could be deployed in multi-player scenarios where groups are challenged to deal with effects between player modeled opponents or utilized to support team-fights between human players, as equivalent reinforcements. Additionally, we plan to construct a one-dimensional proficiency metric that maps DPBM configurations to estimated competence in a game, in order to offer players more unique DDA encounters stemming from different players with similar proficiency. Using the considerably large data set recorded in this study, we seek to benchmark several alternative machine learning techniques as core mechanisms for the underlying player modeling (e.g. recurrent, deep belief, GAN or context-driven LSTM networks), to be able to give practical statements about applicability concerning temporal requirements and resulting accuracy. Furthermore, we envision DPBM as an effective instrument for elevating autonomous game testing and balancing, since actual, precise player behavior can be simulated, and temporary substitutions or continuations of disconnected players in online matches can be facilitated to minimize game experience disruptions.

## CONCLUSION

We presented the design and implementation of an adaptive instance dungeon in the MMORPG *Aion* to evaluate a novel, implicit take on *Dynamic Difficulty Adjustment* that is not dependent on manually composed parameter tuning, but affords a continually adapting challenge through *Deep Player Behavior Modeling*. In an extensive medium- to long-term study ( $n = 171$  over the course of four weeks) we contrasted an opponent applying DPBM to traditional DDA parameter tuning and can report significantly higher intrinsic motivation stemming from the unique game experience of being confronted with strategic behaviors that mirror one's own patterns. Qualitative statements reinforce the approval and positive experience of DPBM, while the consistent and dominant usage of the instance throughout the whole study period reflects its potential to elevate long-term motivation and commitment. Regarding the technical applicability of the approach, we report on the DPBM architecture, its accuracy and data structure and give an estimation about the temporal demand, yielding real-time potential.

According to the guidelines of transparent statistics, the collected data of this approach, as well as its implementation, will be made openly available upon publication, using an open-access repository.

## ACKNOWLEDGMENTS

We would like to thank all participants, testers, *Beyond Aion* for hosting the instance dungeon and Philipp Krüger for his collaboration, advice and porting. This work was funded by the German Research Foundation (DFG) as part of Collaborative Research Center (SFB) 1320 EASE - Everyday Activity Science and Engineering, University of Bremen (<http://www.easecrc.org/>), subproject H2.

## REFERENCES

- [1] Nintendo Research Development 4. 1985. *Super Mario Bros.* Game [NES]. (13 September 1985). Nintendo, Kyōto, Japan.
- [2] Ernest Adams. 2002. Balancing Games with Positive Feedback. *Gamasutra. com*, January 4 (2002).
- [3] Dennis Ang and Alex Mitchell. 2017. Comparing Effects of Dynamic Difficulty Adjustment Systems on Video Game Experience. In *Proceedings of the Annual Symposium on Computer-Human Interaction in Play*. ACM, 317–327.
- [4] Dennis Ang and Alex Mitchell. 2019. Representation and Frequency of Player Choice in Player-Oriented Dynamic Difficulty Adjustment Systems. In *Proceedings of the Annual Symposium on Computer-Human Interaction in Play*. ACM, 589–600.
- [5] ArenaNet. 2005. *Guild Wars*. Game [PC]. (28 April 2005). ArenaNet, Bellevue, WA. Played 2019.
- [6] BioWare. 2012. *Mass Effect 3*. Game [PC,XBox360,PS3,WiiU]. (6 March 2012). BioWare, Edmonton, Kanada.
- [7] Capcom Production Studio 4. 2005. *Resident Evil 4*. Game [Gamecube]. (11 January 2005).
- [8] Thomas Constant and Guillaume Levieux. 2019. Dynamic Difficulty Adjustment Impact on Players' Confidence. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, 463.
- [9] Nevermind Creations. 2019. *Korona:Nemesis*. Game [PC]. (18 August 2019).
- [10] Mihaly Csikszentmihalyi. 2013. *Flow: The psychology of happiness*. Random House.
- [11] Thomas Debeauvais. 2016. *Challenge and retention in games*. Ph.D. Dissertation. UC Irvine.
- [12] Edward L Deci, Richard Koestner, and Richard M Ryan. 1999. A meta-analytic review of experiments examining the effects of extrinsic rewards on intrinsic motivation. *Psychological bulletin* 125, 6 (1999), 627.
- [13] Anders Drachen, Alessandro Canossa, and Georgios N Yannakakis. 2009. Player modeling using self-organization in Tomb Raider: Underworld. In *2009 IEEE symposium on computational intelligence and games*. IEEE, 1–8.
- [14] Anders Drachen, Rafet Sifa, Christian Bauckhage, and Christian Thurau. 2012. Guns, swords and data: Clustering of player behavior in computer games in the wild. In *2012 IEEE conference on Computational Intelligence and Games (CIG)*. IEEE, 163–170.
- [15] Nintendo EAD. 1987. *Zelda II: The Adventure of Link*. Game [NES]. (14 January 1987). Nintendo EAD, Kyoto, Japan.
- [16] Nintendo EAD. 2014. *Mario Kart 8*. Game [WiiU,Switch]. (29 May 2014). Nintendo EAD, Kyoto, Japan. Played 2019.
- [17] William Rao Fernandes and Guillaume Levieux. 2019.  $\delta$ -logit: Dynamic Difficulty Adjustment Using Few Data Points. In *Joint International Conference on Entertainment Computing and Serious Games*. Springer, 158–171.
- [18] Julian Frommel, Fabian Fischbach, Katja Rogers, and Michael Weber. 2018. Emotion-based Dynamic Difficulty Adjustment Using Parameterized Difficulty and Self-Reports of Emotion. In *Proceedings of the 2018 Annual Symposium on Computer-Human Interaction in Play*. ACM, 163–171.
- [19] Christoffer Holmgård, Antonios Liapis, Julian Togelius, and Georgios N Yannakakis. 2014a. Evolving personas for player decision modeling. In *2014 IEEE Conference on Computational Intelligence and Games*. IEEE, 1–8.
- [20] Christoffer Holmgård, Antonios Liapis, Julian Togelius, and Georgios N Yannakakis. 2014b. Generative agents for player decision modeling in games.. In *FDG*. Citeseer.

- [21] Christoffer Holmgård, Julian Togelius, and Georgios N Yannakakis. 2013. Decision making styles as deviation from rational action: A super mario case study. In *Ninth Artificial Intelligence and Interactive Digital Entertainment Conference*.
- [22] Dayana Hristova. 2017. Dynamic difficulty adjustment (DDA) in first person shooter (FPS) games. (2017).
- [23] Robin Hunnicke. 2005. The case for dynamic difficulty adjustment in games. In *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology*. ACM, 429–433.
- [24] Robin Hunnicke and Vernell Chapman. 2004. AI for Dynamic Difficulty Adjustment in Games.
- [25] Changchun Liu, Pramila Agrawal, Nilanjan Sarkar, and Shuo Chen. 2009. Dynamic difficulty adjustment in computer games through real-time anxiety-based affective feedback. *Int. Jnl. of Human-Computer Interaction* 25, 6 (2009), 506–529.
- [26] Ricardo Lopes, Ken Hilf, Luke Jayapalan, and Rafael Bidarra. 2013. Mobile adaptive procedural content generation. In *Proceedings of the fourth workshop on Procedural Content Generation in Games (PCG 2013), Chania, Crete, Greece*.
- [27] Tobias Mahlmann, Anders Drachen, Julian Togelius, Alessandro Canossa, and Georgios N Yannakakis. 2010. Predicting player behavior in tomb raider: Underworld. In *Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games*. IEEE, 178–185.
- [28] Philipp Mayring. 2010. Qualitative inhaltsanalyse. In *Handbuch qualitative Forschung in der Psychologie*. Springer, 601–613.
- [29] Edward McAuley, Terry Duncan, and Vance V Tammen. 1989. Psychometric properties of the Intrinsic Motivation Inventory in a competitive sport setting: A confirmatory factor analysis. *Research quarterly for exercise and sport* 60, 1 (1989), 48–58.
- [30] Olana Missura and Thomas Gärtner. 2009. Player Modeling for Intelligent Difficulty Adjustment. In *Discovery Science*, João Gama, Vítor Santos Costa, Alípio Mário Jorge, and Pavel B. Brazdil (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 197–211.
- [31] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, and others. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529.
- [32] Fausto Mourato, Fernando Birra, and Manuel Próspero dos Santos. 2014. Difficulty in action based challenges: success prediction, players' strategies and profiling. In *Proceedings of the 11th Conference on Advances in Computer Entertainment Technology*. ACM, 9.
- [33] NCsoft. 2003. *Lineage 2*. Game [PC]. (1 October 2003). NCSoft, Seongnam, South Korea.
- [34] NCsoft. 2008. *Aion*. Game [PC]. (25 September 2008). NCSoft, Seongnam, South Korea. Played August 2019.
- [35] Pedro A Nogueira, Vasco Torres, Rui Rodrigues, Eugénio Oliveira, and Lennart E Nacke. 2016. Vanishing scares: biofeedback modulation of affective player experiences in a procedural horror game. *Journal on Multimodal User Interfaces* 10, 1 (2016), 31–62.
- [36] Juan Ortega, Noor Shaker, Julian Togelius, and Georgios N Yannakakis. 2013. Imitating human playing styles in super mario bros. *Entertainment Computing* 4, 2 (2013), 93–104.
- [37] Johannes Pfau, Jan David Smeddinck, Ioannis Bikas, and Rainer Malaka. 2020. Bot or not? User Perceptions of Player Substitution with Deep Player Behavior Models. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. ACM.
- [38] Johannes Pfau, Jan David Smeddinck, and Rainer Malaka. 2018. Towards Deep Player Behavior Models in MMORPGs. In *Annual Symp. on Computer-Human Interaction in Play Ext. Abstracts (CHI PLAY '18)*. ACM, New York, NY, USA, 381–92.
- [39] Johannes Pfau, Jan David Smeddinck, and Rainer Malaka. 2019. Deep Player Behavior Models: Evaluating a Novel Take on Dynamic Difficulty Adjustment. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, LBW0171.
- [40] Mike Preuss, Thomas Pfeiffer, Vanessa Volz, and Nicolas Pflanzl. 2018. Integrated Balancing of an RTS Game: Case Study and Toolbox Refinement. In *2018 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 1–8.
- [41] Andrew K Przybylski, C Scott Rigby, and Richard M Ryan. 2010. A motivational model of video game engagement. *Review of general psychology* 14, 2 (2010), 154–166.
- [42] Nintendo RD1. 2002. *Metroid Fusion*. Game [GBA]. (18 November 2002). Nintendo RD1, Kyoto, Japan.
- [43] Andrew Rollings and Ernest Adams. 2003. *Andrew Rollings and Ernest Adams on game design*. New Riders.
- [44] Robert Rosenthal and Kermit L Fode. 1963. The effect of experimenter bias on the performance of the albino rat. *Behavioral Science* 8, 3 (1963), 183–189.
- [45] Richard M Ryan and Edward L Deci. 2000. Intrinsic and extrinsic motivations: Classic definitions and new directions. *Contemporary educational psychology* 25, 1 (2000), 54–67.
- [46] Lingdao Sha, Souju He, Junping Wang, Jiajian Yang, Yuan Gao, Yidan Zhang, and Xinrui Yu. 2010. Creating appropriate challenge level game opponent by the use of dynamic difficulty adjustment. In *2010 Sixth International Conference on Natural Computation*, Vol. 8. IEEE, 3897–3901.

- [47] Noor Shaker, Julian Togelius, and Georgios N Yannakakis. 2016. The experience-driven perspective. In *Procedural Content Generation in Games*. Springer, 181–194.
- [48] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, and Laurent Sifre et al. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529 (2016), 484–489.
- [49] Adam M. Smith, Chris Lewis, Kenneth Hullet, Gillian Smith, and Anne Sullivan. 2011. An Inclusive View of Player Modeling. In *Proceedings of the 6th International Conference on Foundations of Digital Games (FDG '11)*. ACM, New York, NY, USA, 301–303.
- [50] David Stammer, Tobias Günther, and Mike Preuss. 2015. Player-adaptive spelunky level generation. In *2015 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 130–137.
- [51] Alexander Streicher and Jan D. Smeddinck. 2016. Personalized and Adaptive Serious Games. In *Entertainment Computing and Serious Games*, Ralf Dörner, Stefan Göbel, Michael Kickmeier-Rust, Maic Masuch, and Katharina Zweig (Eds.). Lecture Notes in Computer Science, Vol. 9970. Springer International Publishing, Cham, 332–377.
- [52] Mirko Suznjevic and Maja Matijasevic. 2010. Why MMORPG players do what they do: relating motivations to action categories. *International Journal of Advanced Media and Communication* 4, 4 (2010), 405–424.
- [53] Marco Tamassia, William Raffe, Rafet Sifa, Anders Drachen, Fabio Zambetta, and Michael Hitchens. 2016. Predicting player churn in destiny: A hidden markov models approach to predicting player departure in a major online game. In *2016 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 1–8.
- [54] Gerald Tesauro. 1994. TD-Gammon, a self-teaching backgammon program, achieves master-level play. *Neural computation* 6, 2 (1994), 215–219.
- [55] Valve. 2008. *Left 4 Dead*. Game [PC]. (18 November 2008). Valve, Bellevue, WA, USA. Played 2017.
- [56] Hao Wang and Chuen-Tsai Sun. 2011. Game reward systems: Gaming experiences and social meanings.. In *DiGRA Conference*, Vol. 114.
- [57] Su Xue, Meng Wu, John Kolen, Navid Aghdaie, and Kazi A Zaman. 2017. Dynamic difficulty adjustment for maximized engagement in digital games. In *Proceedings of the 26th International Conference on World Wide Web Companion*. International World Wide Web Conferences Steering Committee, 465–471.
- [58] Georgios N Yannakakis, Pieter Spronck, Daniele Loiacono, and Elisabeth André. 2013. Player modeling. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

## Review Article

# Dynamic Difficulty Adjustment (DDA) in Computer Games: A Review

Mohammad Zohaib 

*Department of Computer Science, BMS College of Engineering, Bangalore 560 019, Karnataka, India*

Correspondence should be addressed to Mohammad Zohaib; zohaib27may@gmail.com

Received 31 May 2018; Accepted 14 October 2018; Published 1 November 2018

Academic Editor: Hideyuki Nakanishi

Copyright © 2018 Mohammad Zohaib. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Dynamic difficulty adjustment (DDA) is a method of automatically modifying a game's features, behaviors, and scenarios in real-time, depending on the player's skill, so that the player, when the game is very simple, does not feel bored or frustrated, when it is very difficult. The intent of the DDA is to keep the player engrossed till the end and to provide him/her with a challenging experience. In traditional games, difficulty levels increase linearly or stepwise during the course of the game. The features such as frequency, starting levels, or rates can be set only at the beginning of the game by choosing a level of difficulty. This can, however, result in a negative experience for players as they try to map a predecided learning curve. DDA attempts to solve this problem by presenting a customized solution for the gamers. This paper provides a review of the current approaches to DDA.

## 1. Introduction

The concept of the video game is continuously changing. The early games like Computer Space and Pong of the early seventies were limited to commercial arcades, but now they are seen in multiple platforms such as cell phones, tablets, computers, and other devices. People are spending in excess of 3 billion hours weekly on gaming [1], which goes to show the extent of change it has brought to our lives.

Entertainment is but one aspect; games are now moving into reality, and the invisible boundaries separating games and reality are now becoming increasingly obscure [2]. Video games now extend to realms of healthcare [3] and education [4]. Experts have studied methods to assess how playing video games affect motor learning and its scope of improving patient involvement with therapy, especially commercial games which could be linked with specialized controls [5].

Although technology in gaming continues to evolve, a general discontent of players with the existing games has been observed due to their limitations in offering challenge levels to suit individual traits of the player like dexterity, learning and adapting ability, and emotional characteristics [6, 7]. Static levels of difficulty that are selected manually

can no longer avoid boredom in players as they, in all probability, would be unable to decide on the challenge level that matches their abilities [8]. Also, constantly calling out the players to select the difficulty levels could distract them and interrupt the game [9]. The fun factor in games depends on three factors: challenge, fantasy, and curiosity [10]. Creating an adequate level of challenge is not easy when players with varying skills are pitted against each other. When an opponent is beaten effortlessly, the game appears boring. Again, in the face of a vastly superior opponent, the game turns frustrating. These two extremes lessen fun, since an optimal challenge is not offered. Csikszentmihalyi [11] first proposed that players, when kept away from the states of boredom or frustration, travel through a “flow channel” (Figure 1) and this was incorporated into a gaming scenario by Koster [8].

This model indicates how the difficulty of a task directly relates to the performer's perception. The flow channel shows that the difficulty level can be gradually enhanced, as sufficient time exists for the players for learning and improvement to meet this challenge [11]. Thus, the model prevents the frustration of difficult situations and the boredom of simple ones. In a different study, Malone [10] suggested that if the fantasy, challenge, curiosity, and control in games could be

TABLE 1: DDA research studies since 2009.

| Year | Journal Papers | Conference Papers | Theses | Books | Total |
|------|----------------|-------------------|--------|-------|-------|
| 2009 | 1              | 2                 | 1      |       | 4     |
| 2010 | 1              | 7                 | 1      |       | 9     |
| 2011 | 2              | 5                 |        | 1     | 8     |
| 2012 | 3              | 8                 | 2      |       | 13    |
| 2013 | 2              | 5                 | 3      | 1     | 11    |
| 2014 | 1              | 4                 | 1      | 1     | 7     |
| 2015 | 1              | 5                 | 1      |       | 7     |
| 2016 | 3              | 5                 | 3      |       | 11    |
| 2017 | 5              | 7                 | 2      |       | 14    |
| 2018 | 0              | 0                 | 0      | 0     | 0     |

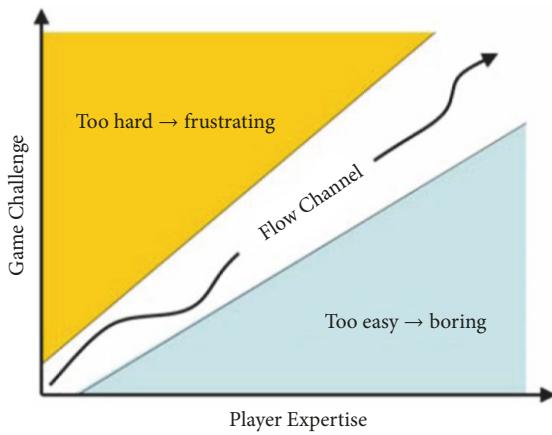


FIGURE 1: Flow channel concept proposed by Csikszentmihalyi.

balanced and associated with the gradual enhancement of difficulty level stated earlier, it could be possible that the ensuing game could keep the player entertained. Peeters [12] in her study suggested designing an automated platform for scenario-based training so that learners could engage in personalized autonomous training where agent-based notions such as beliefs, desires, and intentions can be used to deal with the gamer's competency and skills.

Numerous studies have been addressing the problems of static levels and have proposed the dynamic difficulty adjustment (DDA) technique that allows the automatic mapping of playing experience with the individual skills. DDA is a technique of automatic real-time adjustment of scenarios, parameters, and behaviors in video games, which follows the player's skill and keeps them from boredom (when the game is too easy) or frustration (when the game is too difficult). The essence of the DDA is to retain the interest of the user throughout the game and to offer a satisfactory challenge level for the player [13]. Andrade et al. suggested that DDA must cater the following three basic needs of games [14]:

- (1) The game needs to automatically track the player ability and rapidly adapt to it
- (2) The game must follow the player's improving or falling level and maintain a balance in accordance with the player's skill

(3) The process of adaptation must not be clearly perceived by the players, and successive game states need to have coherence with the earlier ones

Before applying the DDA, an understanding of the term "difficulty" is necessary. Though abstract, certain aspects need to be considered to assess and measure difficulty. Some of them are characteristics of design [15], number of resources [16], number of losses or victories [17], and so on. Nevertheless, DDA is not as easy as merely giving a player some healthier items in times of trouble. It needs an estimate of time and an entry at the right instant, as keeping the player absorbed is complicated in an interactive sense [16].

## 2. DDA Studies in the Last Ten Years

After 2009, there have been many research studies related to methods to develop or improve DDAs, including innovative applications in diverse fields. It is notable that the number of research papers in 2012 and 2017 is almost three times the number of research papers presented in 2009 (Table 1).

In this study, we have focused on DDA research studies undertaken after 2009 and have presented the important categories observed over the last decade (2009–2018). Going by the data presented in Table 1, we observe that, in the last decade, there has been a significant increase in the number of research papers on DDA over the years, and it was the highest in 2017.

Figure 2 depicts the DDA research studies carried out in the last ten years, including journal and conference papers, thesis work, and book chapters for every year.

## 3. Classification of DDA Approaches

Various methods for DDA are proposed in the literature (Table 2). The one common aspect in all methods is a requirement to measure (in a manner that may be implicit or explicit) the level of difficulty being faced by the player at any given instant. These measures are estimated by heuristic functions, also called **challenge functions**.

They assign a value for any game state that is indicative of the difficulty level of the game felt by the player at any given moment. Typical examples of heuristics in use are success rates of hits, numbers of pieces won and lost, life points,

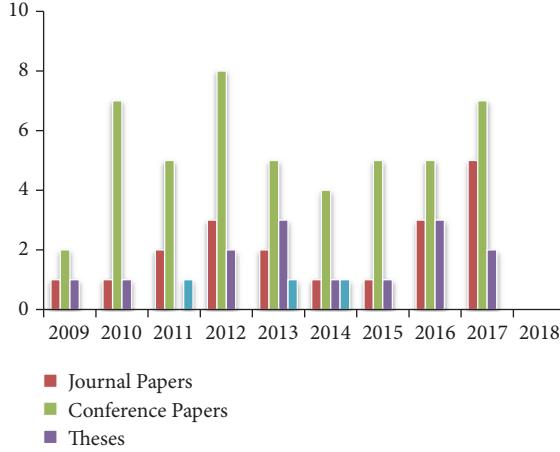


FIGURE 2: DDA studies over the past decade.

TABLE 2: List of DDA approaches.

| Author(s)                          | Approach  |
|------------------------------------|---|
| Xue et al.                         | Probabilistic Methods   |
| Pedersen, Togelius, and Yannakakis | Single and multi-layered perceptrons                            |
| Spronck et al.                     | Dynamic scripting   |
| Hunicke and Chapman                | Hamlet System   |
| Hagelback and Johansson            | Reinforcement Learning  |
| Li et al.                          | Upper Confidence Bound for Trees and Artificial Neural Networks |
| Ebrahimi and Akbarzadeh-T          | Self-organizing System and Artificial Neural Networks           |

completion time for assigned tasks, or any other metrics for calculating game scores.

There are several ways we can classify approaches to DDA.

**3.1. Probabilistic Methods.** A study on a framework that sees DDA as a problem of optimization was carried out [18]. This approach maximized player engagement all through the game. They modeled the progression of the player on a probabilistic graph (Figure 3) that maximized engagement as a well-defined objective function.

A dynamic programming technique having high efficiency was utilized to solve it. They assessed the DDA implementation using a mobile game by Electronic Arts, Inc. The group treated by DDA showed a clear increase of core engagement metrics, e.g., total number of plays and duration of game, while being revenue neutral when evaluated with the control group that did not have enabled DDA. This framework can be extended to a variety of game genres. DDA can be successfully applied to other genres if an appropriate progression model is constructed. The states for level-based games can be established by two important facets: trial and level. For games that are more complex having multiple or nonlinear progressions (e.g., role-play games), too, the states having varied dimensions can be defined. The graph would

then be more complex since more states and links would be included.

Segundo et al. [19] proposed the creation of a parameter manipulating method for DDA, which aims to enhance the pleasure of gaming. The proposed method utilizes probabilistic calculations that could be deployed in a challenge function. A sample of students was provided with a questionnaire to assess whether a significant statistical difference existed in the understanding of game difficulty, game play, and the desire to play often with and without the method. The results indicated that the DDA version showed better results than the other versions with regard to game play and the desire to play often.

In a study [20], it was proposed that both online and offline learning techniques could be used for DDA. In the offline learning, a genetic algorithm was applied to create a fuzzy rulebase for game tactics during play to manipulate the computer-controlled adversaries. In the online learning, a probabilistic method was used for adapting the game strategies to the player. The level of difficulty of the game can be adjusted in accordance with the preference of the player seeking a challenge. The results demonstrated the superior capability of the evolved offline rulebases and the efficacy of the suggested online learning method for DDA.

Bunian et al. [21] developed a modeling technique by use of data gathered from players involved in a Role-Playing Game (RPG). The proposed technique has 2 features: (i) a player's Hidden Markov Model (HMM) tracking in-game traits for modelling individual differences and (ii) use of the HMM output to generate features of behaviors for classifying real-world characteristics of players that include expertise along with the big five personality traits. The results showed the prediction capability for some of personality traits, like conscientiousness and expertise. A logistic regression model was trained considering the composition of the freshly created behavioral features for 66 participants. A three-fold cross validation was used as the dataset was small. The prediction accuracy for conscientiousness and expertise category was 59.1% and 70.13%, respectively.

Bayesian optimization techniques were used in a study [22] to design games which maximize the engagement of users. Participants were paid to attempt a game for a short period, following which they could continue to play without payment or quit voluntarily. Engagement was measured by their persistence, estimates of duration of other players, and a survey after the game. Utilizing Gaussian surrogate-based process optimization, experiments were conducted to establish game design features, especially those affecting difficulty leading to maximum engagement. The converging outcomes indicated that overt difficulty manipulations were effectual in modifying engagement only with the covert manipulations, demonstrating the user's self-perception of skill as being critical.

Hintze, Olson, and Lehman [23] proposed the idea of orthogonal coevolution and verified its effectiveness in a game that was browser-based modified from a scientific simulation. The outcomes demonstrated that evolving adversaries together with evolved friends could lead to seamless DDA and permit gamers to experience more diverse

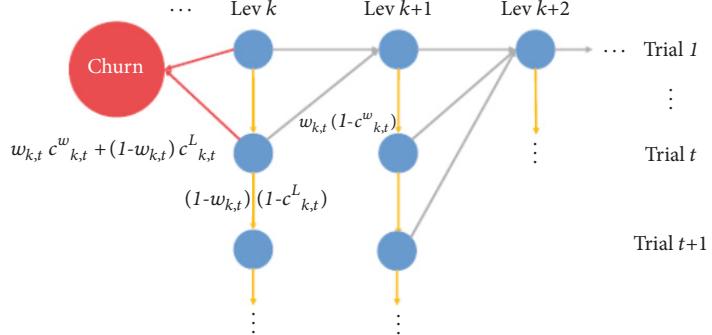


FIGURE 3: Probabilistic graph showing the player’s progression model in a typical level-based game.

situations. They concluded that such an orthogonal coevolution could be of promise for adjusting gaming difficulties.

**3.2. Single and Multilayered Perceptrons.** In a study by Pedersen, Togelius, and Yannakakis [24], the relationship between parameters of level design of platform games, player experience, and individual characteristics of play were studied. The studied design parameters had relation to the size and placement of level gaps and the presence of direction changes; and the constituents of a player’s experience comprised frustration, fun, and challenge. A neural network model, which mapped between characteristics of playing behavior, design parameters of levels, and player emotions, was trained utilizing game session data and evolutionary preference learning.

Data was gathered from the Internet. Users were inducted through messages on mailing lists and blogs and sent to a web page which contained a Java applet initiating the game and a questionnaire. After playing the games and completing the questionnaire, all the characteristics (gameplay, controllable, and player experience) were recorded in a repository on a server. After analyzing this data, they attempted a function approximation based on gameplay and controllable characteristics to record emotional choices utilizing neuroevolutionary preference learning. This data representing the function was full of noise, since the choices of the players were highly subjective and the style of playing varied. All of this, coupled with the meager training data amount, suggests the usage of a function approximator that is robust. An artificial neural network (ANN), being a nonlinear function, is a suitable option for the approximation in mapping between data and reported emotions. Therefore, a simple single-neuron (perceptron) was used to learn the relationship between characteristics (ANN data input) and the analyzed emotional choice. The primary purpose for the use of a single neuron rather than a multilayered perceptron (MLP) here was that the trained function approximator needed to be analyzed. Though MLP can approximate the function more accurately, it is simpler for us to visualize the derived function when presented by a single-neuron ANN. Learning was obtained by artificial evolution by adopting the preference learning method [25]. A generational genetic algorithm was deployed, utilizing a goodness-of-fit

function which measured the variation between the recorded emotional preferences and the corresponding model output. Results showed that there was high accuracy of prediction of challenge (77.77%), frustration (88.66%), and fun (69.18%) using a single-neuron model, which recommends using more elaborate nonlinear approximators. The study also discussed how the models generated could be used to generate game levels automatically, which would improve the player experience.

In another study, Shaker, Yannakakis, and Togelius [26] demonstrated the automatic generation of personalized levels for platform games. They built their model on the earlier work by Pedersen, Togelius, and Yannakakis [24]. At first, single layer perceptrons (SLP) were used to approximately evaluate the affective level of the players. The input subsets were selected by the sequential feature selection. To generate content customized to suit real-time player experience automatically in real-time, predicting emotions, to some extent, from controllable features is necessary. For this, the rest of the controllable features not already in the chosen feature subset were forcibly entered in the input of the multiple layer perceptron models, and the topology of the networks was made optimal for the highest accuracy of prediction.

In this study, dynamic adaptation to changes in playing styles was assessed. The model’s capacity to generalize over players of various types was tested. To carry this out, two artificial intelligence (AI) agents were deployed for play in turns, while tracking the growth of the fun value. The experiment commenced from a level generated at random. The agents played 100 levels with an agent switch after every 20 levels. The result showing the variation in fun level across 100 levels is shown in Figure 4.

It is seen that the fun value is about 70% for the initial 20 levels when the first agent plays, increases to 80% when the next agent plays for 20 levels, and drops down to 70% when the first is brought back to the game. It clearly shows the model’s capability to adjust to the player type. As a further test, the same trial was repeated on 4 human players in a reduced set of 12 levels. The result of this trial is illustrated in Figure 5, which shows the progress of fun over 48 levels. The results are similar to those obtained from the AI agents. It clearly indicates that the model robustly adapts to an individual player generalizing over various kinds of players.

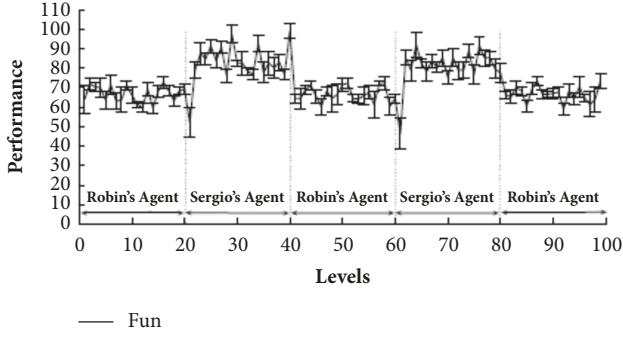


FIGURE 4: Two-agent optimized levels of fun.

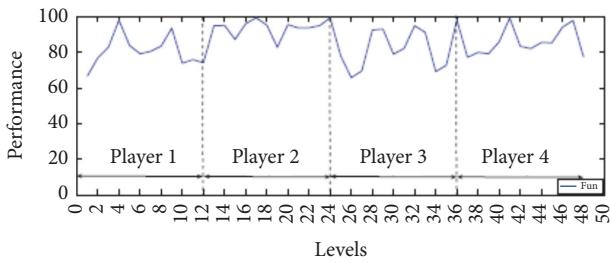


FIGURE 5: Four-player optimized levels of fun.

A study [27] constructed computational models of a player’s experience derived from interaction in gameplay for use as fitness functions for content creation in games. A classic platform game’s modified version was used for their experiments, and player data was collected from the Internet. They used the preference learning method for generating models of player experience. Feature selection was utilized to lower the features in the model. The data from training of nonlinear perceptrons was used to approximate the mapping functions between controllable features and selected gameplay. They presented the results of optimal construction of multilayer perceptrons (MLP) and the MLP model performances. They finally discussed the ways by which induced models could generate game content automatically.

Most DDA methods are based on the intuitions of designers, which do not reflect real-world playing patterns. Therefore, Jennings-Teats, Smith, and Wardrip-Fruin [28] created Polymorph that used methods from machine learning and level generation to analyze player skill and level difficulty, thereby dynamically creating levels in a 2D platformer game having continuously desired challenges. The DDA problem was addressed by generating a machine-learned difficulty model in a 2D platformer game using a model of the existing skill of the player. Multilayer Perceptrons accessed from play traces are used. These traces are gathered using a web-based tool which assigns users with various short-level components and rates them on a difficulty level. The Polymorph model utilizes the models of difficulty to choose the suitable level segment for the existing performance of the player.

Carvalho et al. [29] presented a method for generating gameplay sessions for endless games. This genre still remains largely unexplored in literature. The method uses a four-step process starting from the generation of required content to

placing the content through the gameplay sessions. A robust evaluation technique was also designed. This technique utilizes both features that can be adjusted by a designer and gameplay items gathered from gameplay sessions. The usage of 2 neural networks is a new technique that agrees with the idea of game as a service and supports it throughout the life cycle of the game. The two neural networks have different purposes: the first receives merely features that are controllable as input, and the other receives both non-controllable and controllable features as input. Both the neural networks adjust chunk (fixed-size segments of the game on which gameplay elements are placed) difficulty as their output. Thus, the first network is used in the initial development stages, where one has access to only controllable features of these chunks, and the other is used to periodically adjust the game once it is made available. Both neural networks are Multilayer Perceptrons, each having a hidden layer.

**3.3. Dynamic Scripting.** Dynamic scripting is an online unsupervised learning approach for games. It is computationally rapid, robust, efficient, and effective [30]. It operates many rulebases in the game, running one for every opponent type. These rules are designed manually utilizing domain-specific information. With the creation of a new opponent, the rules that constitute the script guiding the opponents are taken from the rulebase based on their type. The probability of a script rule selection depends on the weight value allotted to the rule. The rulebase adjusts by amending the values, reflecting the rates of failure or success of the related script rules.

In this approach, learning takes place progressively. On completing an encounter, the rule weights used in the encounter are treated depending on their effect on the result. The rules leading to success have their weights increased, while those leading to failure have their weights decreased. The remaining rules are adjusted accordingly so that the sum of all the rulebase weights remains constant. Dynamic scripting is used to generate fresh opponent tactics while increasing the level of difficulty of the game’s AI to match the level of experience of the human player (Figure 6).

There are three different enhancements to this technique allowing the opponents to learn playing a balanced game:

(1) High-fitness penalizing: The weight balancing provides rewards in proportion to the fitness value. To obtain mediocre rather than optimal behavior, the weights can be amended to reward mediocre values of fitness and punish superior values.

(2) Weight clipping: The maximum value of weight decides the highest optimization level that a learned tactic can reach. A high value for the maximum permits the weights to increase to high values, so that soon the most effective rules will nearly always be chosen. This results in scripts having near to optimal values. Similarly, low values for the maximum hamper growth of weights. This creates a large variation in scripts generated, many of which would be nonoptimal. This method automatically varies the maximum value, thereby enforcing a balanced game.

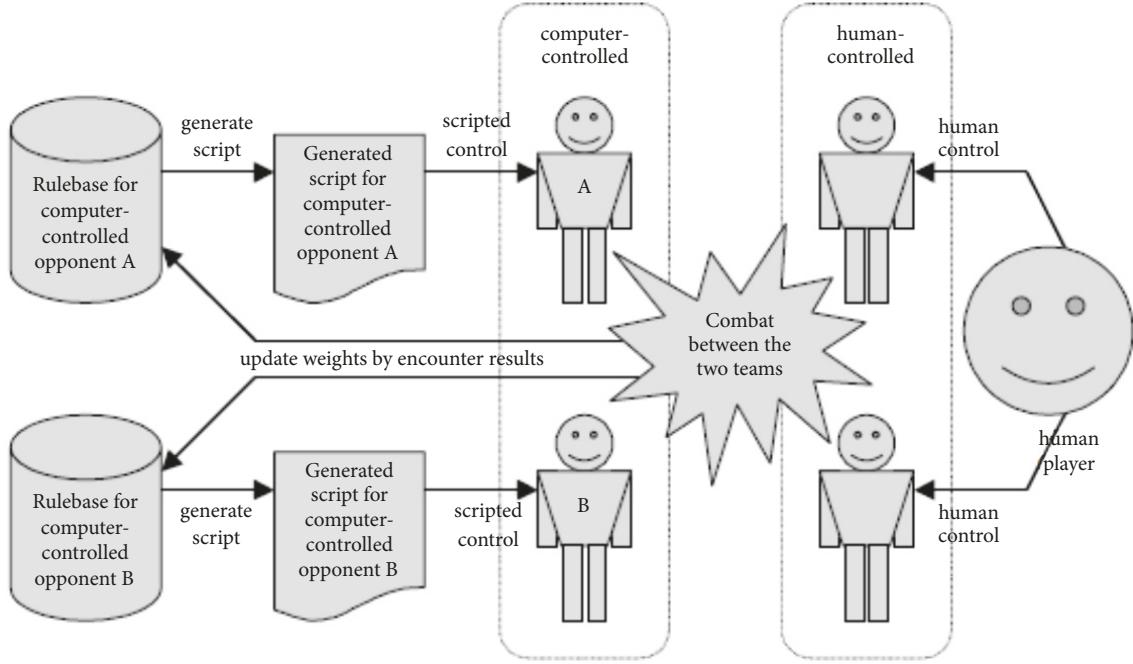


FIGURE 6: The dynamic scripting process.

(3) Top culling: Similar to weight clipping, it uses a similar mechanism for adaptation for the maximum value, the difference being that, here, weights are allowed to grow above the maximum value. However, rules having weights in excess of the maximum value do not get chosen for a generated script. As a result, frequent wins of computer-controlled opponents cause effective rules to be rejected, making opponents use weak tactics. Conversely, frequent losses would cause rules with high weights to become selectable, making opponents use weak tactics.

Experiments conducted by the authors showed that DDA by dynamic scripting is effective. It was also seen that the three approaches were tested; high-fitness penalizing was not successful, but the two other approaches did well.

**3.4. Hamlet System.** Most games use the concept of inventory, i.e., the store of items a player gathers and takes all through the game. The relative amplexness or lack of items in inventory directly impacts the experience of the players. Games are designed to control the exchange of items between the player and the world [31].

These maps of producer-consumer links can be seen as an economy or a dynamic system. Hamlet, a DDA system built by Hunicke and Chapman [13], uses methods taken from Operations Research and Inventory Management. It studies and adjusts supply and demand of the inventory in the game so as to manipulate the game difficulty. The system is essentially a group of libraries maintained in the engine of Half Life. Hamlet has functions in the following:

- (1) Managing game statistics in accordance with statistical metrics defined in advance
- (2) Deciding adjustment tasks and rules
- (3) Carrying out those tasks and rules

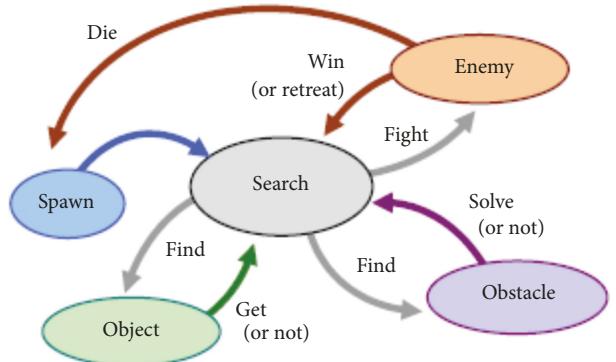


FIGURE 7: A simple FPS game state transition illustration.

- (4) Presenting data and system settings
- (5) Creating traces for playing rounds

Hamlet uses metrics for monitoring incoming game information as the players advance through the game world. It predicts the future state of the players from this information. Whenever an unwanted but preventable state is predicted, Hamlet steps in and tweaks game settings as required. Essentially, it tries to anticipate when the player is struggling repeatedly and nearing a state where his existing resources can no longer meet requirements. When this struggle is detected, Hamlet intervenes to assist the player to continue the game.

Keeping up a player's challenge and interest level is not an easy task. One popular approach is the flow model proposed by Csikszentmihalyi [11]. In an FPS environment (Figure 7), the gameplay can be illustrated with a fairly simple state transition picture.

Players engage in cycles of seeking, fetching, solving, and battle. Each new level creates new foes and hurdles. Difficulty levels and skill get enhanced with time. Hamlet is tailored to keep players within the Csikszentmihalyi's flow channel by promoting some states and demoting others. The basic aim is to keep the players in engaging loops of interaction for durations that are most appropriate based on their skill and experience gathered. To sum up, Hamlet looks to

- (1) Evaluate when adjustments are required
- (2) Decide on the changes
- (3) Make changes seamlessly

When a player is struggling, in many FPS games, it is observed that constant inventory shortfalls occur in locations where the player's existing resources do not meet the required demands. By noting trends in the inventory expenses of players, probable shortfalls are looked for, thereby identifying probable opportunities for adjustment. The evaluation process begins with the establishing of metrics to assess data. The damage data, based on its probability distribution, are analyzed. Inventory theory equations provide a basis for modeling the player's overall inventory and flow. Shortfalls are predicted based on total damage probabilities. Hamlet accordingly takes reactive and proactive action by making adjustments. Adjustment protocols are defined in the Hamlet system. Adjustment actions, together with cost estimations, form adjustment policies.

*3.5. Reinforcement Learning.* Games are played by a variety of players who use varied gaming patterns and strategies. Hence, a static game AI cannot deal with the gaming styles of all kinds of gamers. A game AI that is adaptive, therefore, can create varied gaming experiences for different playing styles and thus add interest and repeatability of play to a game. Such mechanisms have been studied with interest in recent years. For example, evolutionary algorithms by Togelius et al. [32] were used to create racing tracks that were popular with players.

Hagelback and Johansson [33] in a study observed that players enjoy playing an evenly matched game against opponents who adapt to their styles. To this end, Tan, Tan, and Tay [34] developed an adaptive AI for games that promotes even play rather than beating opponents. Here, a dynamic computer controlled opponent adapts its behavior, according to its opponent's moves. This DDA technique uses adaptive AI in the game to adjust game behaviors and parameters in real time automatically in response to the skill of the player. It can keep the player engrossed for longer periods and enhance their experience.

As mentioned, here, DDA is carried out in real time. The adaptive AI of the game requires being adept enough to make unforeseeable but rational judgments like human players but must not display the overtly thoughtless behavior. The AI also needs to be able to correctly assess its opponent in the beginning of the game itself and adjust its playing style to opponent's skill. This study proposed two adaptive algorithms, adaptive unichromosome controller (AUC) and the adaptive duochromosome controller (ADC) that utilized concepts from evolutionary computation and reinforcement learning [34] to adaptively play in real-time. Two metrics,

winning percentage difference ( $|W-L|$  and D to be minimized, where W, L, and D are wins, losses, and draws) and mean score difference ( $|s_1-s_2|$  to be minimized and  $\max(s_1, s_2)$ , where s denotes scores of players 1 and 2), were used as indicators of entertainment value. First, the game is so designed that the AI has the capability to beat the player. Second, the game AI can make deliberate mistakes, termed as artificial stupidity; therefore, the players remain interested in the game.

The training and adaptation of the AUC take place in real time when the game is in session. As its name suggests, AUC stores a single chromosome that maps to seven numbers, one corresponding to each behavior component. Each number indicates the probability of deploying a behavior component when a waypoint is crossed. The expected behavior mapped by this chromosome exemplifies a victory strategy. The chromosome tailors the proficiency of the opponent by mapping a behavior set which would be sufficient to beat him. The chromosome is initialized randomly at the beginning of every game. Whenever a waypoint is crossed, a set of rules updates the chromosome. The assumption here is that the complement of an expected victorious strategy is a losing one. The ADC and AUC are similar except that the former does not assume that the complement of an expected victorious strategy is a losing one. Instead, two sets of chromosomes are maintained, one winning and one losing chromosome, all through the game. The chromosomes are updated by different rules for wins and losses.

These controllers were tested against static controllers of varied driving traits to simulate various styles of play like heuristic controllers, neural network controllers, reverse enabled controllers, predictive fast controllers, etc. The effects of changing the mutation and learning rate were studied for both controllers (algorithms). The pattern of the difference of scores was assessed and both achieved score differences of 4 or less in at least 70.22% of the games. Wins and losses were also well scattered across the sequence of games played consecutively. It was also seen that the AUC had a low memory footprint, and the ADC was capable of maintaining a lesser number of drawn games, which could keep the player interested. The final values of the chromosomes showed that the algorithms choose various combinations of behavior components to deal with different opponents. Both controllers were able to learn proper sets of behavior components for the various opponents by way of winning percentage and mean score. Also, both were able to generalize satisfactorily to different opponents.

Sekhavat [35] suggested a personalized DDA method for a rehab game that manages difficulty settings automatically, based on a real-time patient's skills. Concepts of reinforcement learning were used as a DDA technique. It was shown that DDA has multiple objectives, in which objectives could be evaluated at different times. To solve this issue, it was proposed to use Multiple-Periodic Reinforcement Learning (MPRL) which enables the evaluation of various objectives of DDA in separate time periods. The experiments showed that MPRL performed better than available Multiple-Objective Reinforcement Learning methods in user satisfaction and enhancing the patient motor skills.

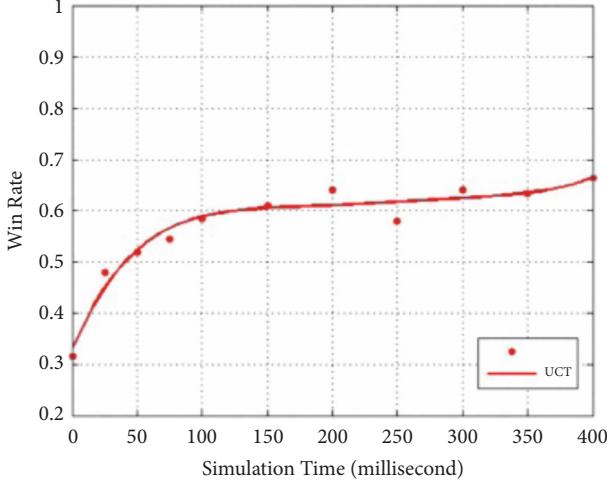


FIGURE 8: DDA from UST.

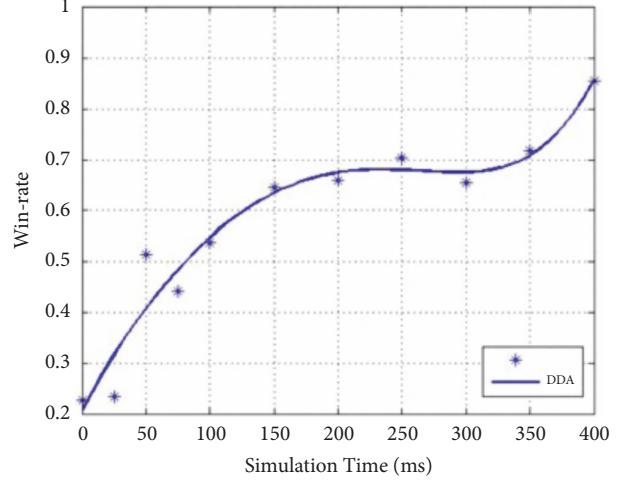


FIGURE 9: DDA from ANN.

**3.6. Upper Confidence Bound for Trees and Artificial Neural Networks.** Li et al. [36] developed a DDA technique using artificial neural networks (ANN) from data derived from the upper confidence bound for trees (UCT). The Pacman game was used as a test-bed for this study. Considering that UCT is a computing intelligence method, UCT performance significantly correlates with the duration of simulation [37]. Figure 8 illustrates DDA process from UCT-created data.

Here, the x-axis denotes simulation time, which is in the range 0–400 ms. The y-axis denotes win-rates of opponents (ghosts) which are in the range 30%–70%. The curve rises steeply in the period 0–100 ms; this period has a higher number of test data. After 100 ms, the curve flattens. The win-rate attains a maximum at 400ms. The reason for the stability of the win-rate is because of UCT being a stochastic simulation approach. In the interval 0–100 ms, the sample space is bigger and the stochastic outcomes become more accurate. Hence, the UCT performance is drastically improved. Also, after crossing 100 ms (threshold value), the accuracy of results is still fairly good so that the win-rate grows smoothly even at higher values of simulation time. UCT can be used as DDA in real-time games, too. By merely adjusting the UCT simulation time, we obtain game opponents of increasing difficulty levels.

ANN can be trained from UCT-created data. Even though the UCT approach can be deployed as DDA for games like Pacman, it is not practical to be used for complex online games because of UCT’s computational intensiveness. But then since UCT’s performance can be tweaked by varying the simulation time, ANN offline training becomes possible by running the UCT-created data with changed simulation times. Thus, DDA can be generated from UCT-created data, bypassing the computational intensiveness. In this study, the 3-Layered Feed-Forward Artificial Neural Network model in WEKA was used for the implementation.

DDA can also be created from ANN. The weights and bias of ANN are reserved in MDB files. Opponents are managed by ANN by loading MDB files.

Figure 9 illustrates the DDA from ANN. The x-axis ranging within 0–40ms is the same as Figure 7. The y-axis represents win-rate of opponents (ghosts) controlled by ANN from data created by UCT with changed simulation time ranging within 20%–86%. The curve rises steeply in the range 0–100ms. After 100ms, the curve rises steadily and the win-rate peaks at 400ms.

The performance of the opponent’s neural network depends on the training data quality. With insufficient incidences for a certain route, the ANN training remains poor. With ample incidences for all routes, the trained ANN performs well. With the increasing growth of simulation time, the UCT data achieve greater precision, which in turn creates ANN that is better trained. Comparing the two curves, we note that the DDA curve tends to rise from a minimum to a maximum simulation time. Hence, a valid DDA curve can be derived by ANN training from UCT based data. Thus, UCT is a good computation intelligence algorithm which performs better when the simulation time increases. It can therefore be used as a DDA tool by tweaking the simulation time. UCT can also create data to train ANN.

A data-driven approach for DDA was proposed by Yin et al. [38]. The objective was to match the player’s performance to the required conditions laid down by the designer. The data pertaining to dynamic game states and in-game player performance were used for taking decisions on adaptation. Trained ANNs were utilized to map the relationship between player performance, dynamic game state, adaptation decisions, and the game difficulty that resulted. The predicted difficulty enables effective adaptation of both magnitude and direction. An experiment on a training game application demonstrated the efficacy and stability of the suggested approach.

**3.7. Self-Organizing System and Artificial Neural Networks.** In another study [39], a self-organizing system (SOS) was developed, which is a group of entities that presents global system traits through local interactions while not having centralized control. This method proposes a new technique that tries to adjust the difficulty level by creating an SOS of

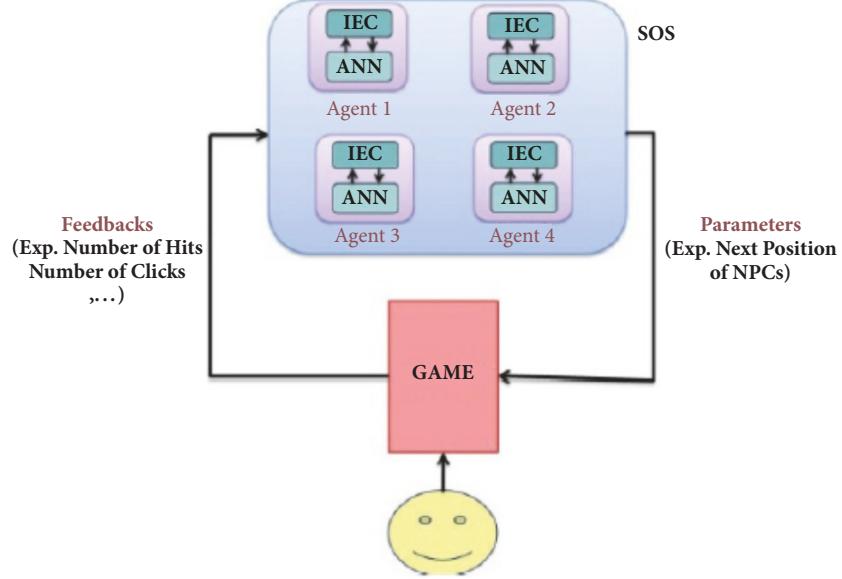


FIGURE 10: Illustration of self-organizing algorithm.

Non-Player Characters (NPCs) that are not in the player's control. To track human player traits, ANNs are used in the system. ANNs need to adapt to players having varied levels of skills and traits; therefore an evolutionary algorithm having adaptation skills was developed which modifies ANN weights (Figure 10).

Pacman game has been used as a test bed. There are two agents in the game: Pacman (the player) and ghost (opponent). The authors have considered four types of Pacmans having varying intelligence levels. The first is Cost-Based Pacman, a local agent who locates his subsequent location based on the position of his nearby ghosts. The second, named Distance-Based Pacman, is a global agent, who considers locations of all ghosts prior to deciding the next move. The third, neither fully global nor local, is named Nearest Distance-Based Pacman. The three Pacmans represent players having varying levels of skill. The fourth, Random Pacman, moves randomly and is not classified in any of these categories.

A neuroevolutionary controller for each Ghost was developed in order that they adapt to the different skill levels of the players. They decide the next position based on environmental precepts. Every ghost was given a feedforward neural network having a concealed layer. The topologies of the networks were decided during the game. Ghosts are first trained offline. This offline training helps to create chromosomes that perform better than random chromosomes. The Cooperative Coevolution Algorithm is used to train ghosts offline [40]. A subpopulation of chromosomes is considered for every ghost, which have neural network connection weights. The best performers in every subgroup are chosen as representatives. Next, to assess chromosomes of each subgroup, a game is arranged between the representatives and these chromosomes. On completion of the game, its fitness is assessed and assigned. This process is repeated till all the

chromosomes of every ghost are mapped. After assessment of all chromosomes in the subgroups is carried out, the genetic algorithm selects, crosses over, and mutates for producing new chromosomes. When the stop requirements are met, this sequence of events halts.

Next, online learning of the controllers takes place. Before the game starts, the required chromosomes are loaded in the ghost controllers. As the skill levels of the players are still unknown, intermediate chromosomes are selected for all ghosts. The subgroups are sorted on the basis of individual fitness; the median chromosomes are selected. The game begins when the chromosomes are loaded in the neural networks. The game runs for a short duration after which the system fitness is assessed. Neural controllers are trained using the Interactive Evolutionary Computation (IEC) where the fitness function replaces human evaluation [41]. As human evaluation results in fatigue, IEC optimizes systems effectively. System fitness is indirectly assessed using player feedback, e.g., number of keys pressed, occasions of key switches, and Pacman's wall hits. After each duration, these numbers are analyzed to assess fitness.

The results show that this system is capable of adapting to many skill levels by selecting proper factors that hasten convergence to the optimal requirements.

A study [42] explored the use of NEAT and rtNEAT neuroevolution techniques to create intelligent adversaries for games having real-time strategies. The primary objective is to convert the challenge created by the adversaries to match the recompetence of the player in a real-time situation, thereby resulting in a greater entertainment value experienced by the player. The study introduced the application of the neuroevolution techniques to Globulation 2 (G2), real-time strategy game for DDA. Initially, NEAT was used to optimize the functioning of G2 nonplayer characters and two suggested challenge factors were investigated by offline trials

in G2. The results indicated that the aggressiveness factors and warrior numbers are contributors to challenge because neuroevolved agents obtained succeeded in outperforming all typical AI nonplayer characters available for playing the game.

**3.8. Affective Modeling Using EEG.** Earlier researchers studied heuristic approaches based on a game state. Generally, in a game that tracks an ongoing score, decisions on DDA application can be taken when the difference between the scores of the players exceeds a threshold value, i.e., when one player becomes stronger than the other, and assuming that this would result in boredom in the stronger and frustration in the weaker player.

Stein et al. [43] have proposed a different method—measuring the excitement of players and setting in motion the game levels when the level of excitement dips below a threshold value. They have attempted to address the main issue of gaming experience directly, rather than depending on heuristic scores to decide when they are bored with the game.

An affective-state regulation technique was implemented by using headsets to decipher electroencephalography (EEG) signals and the mechanism to modify the signal to an affective state.

Next, assessing this affective state, DDA is deployed by the game. Two studies were conducted. In the first, the relationship between the EEG signals and game events (GE) was investigated. The results showed a significant correlation between the indicator for short term excitement (STE) and GE. Playing experiences were attempted to be enhanced by maximizing STE. In the second study, this EEG-initiated DDA was compared to (1) a typical heuristic technique which used elapsed duration and game status and (2) a control game without DDA. A case study was presented for the EEG-initiated DDA approached in a customized version of the Boot Camp game. The study confirmed that (1) players preferred the EEG-initiated DDA to the two other choices and (2) this method greatly increased the excitement level of the players. The study also indicated that the option of the initiating strategy is significant and greatly impacts experience of the players.

Afergan, Mikami, and Kondo [44] used functional near-infrared spectroscopy for collecting passive brain-sensing information and detecting long durations of overload or boredom. Using these physiological signals, a simulation was adapted for optimizing real-time workload that permits the system to adjust the task for the user at each moment in a better manner. To demonstrate this concept, they conducted laboratory experiments where participants, in a simulation, were assigned path planning for several unmanned aerial vehicles. The task difficulty was varied based on their state by the addition or deletion UAVs and they found that errors could be decreased by 35% over and above a baseline level. The results indicated that fNIRS brain sensing can be used to detect real-time task difficulty and an interface constructed that enhances user performance by DDA.

Fernandez et al. [45] adapted the levels of difficulty of a basic 2D platform game, working on and building levels

automatically. The method proposed consisted of DDA and Rhythm-Group Theory, a procedural content development approach, along with attention levels gathered from EEG data. Trials were planned in a manner that players needed to perform 5 varied levels automatically created by their performances and EEG data collected through a biosensor during play. Results indicated that the method adapted successfully to the difficulty levels as per the status of the player. Additionally, the method calculated difficulty utilizing calculated real-time values to decide the level.

## 4. Future Work

There are many opportunities for higher research into DDA by creation of level structures. Researchers can go beyond the standard 2D platformer video game genre and apply the same DDA concepts to different genres. Also, more research is required in new search-based techniques for identifying optimal levels. Researching player models other than agent types could be another promising area for more research. As the fitness function is a crucial element in game design, increasing its complexity by the addition of more variables that could consider many other aspects of play is yet another promising area.

An interesting research could be to investigate the possibility of covering traits like playing style. The concept of mapping the human player and developing a player model accordingly is yet another possibility. A player model that includes more behavioral aspects could yield interesting observations.

Many player modeling techniques exist currently. Integrating a few of these with present DDA approaches could open up some possibilities of interest and yield more DDA techniques tailored to a gamer's preference.

## 5. Conclusions

DDA techniques have been proven in the literature to be useful tools for incorporation in complex and dynamic systems. This investigation has presented a review on DDA applications and directions in many diverse kinds of games in the past decade highlighting some of the most representative types for every application. There are numerous application studies of DDAs in various domains, including generalizations and extensions of DDAs. The number of approaches presented here is neither complete nor exhaustive but merely a sample that demonstrates the usefulness and possible applications of AI techniques in modern video games.

## Conflicts of Interest

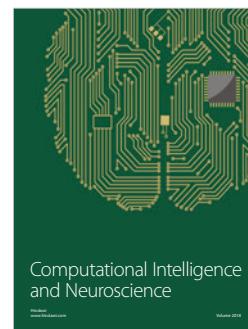
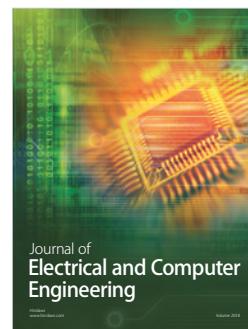
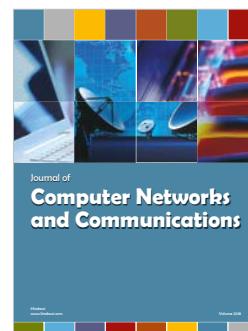
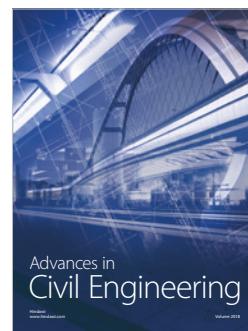
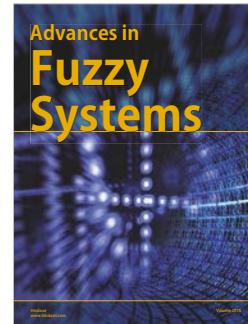
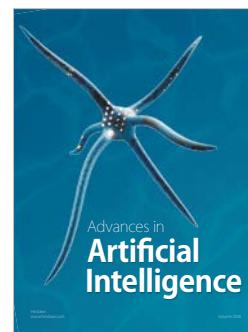
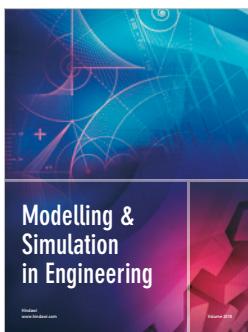
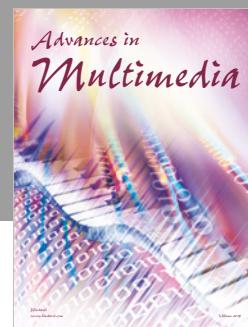
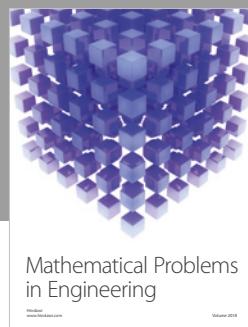
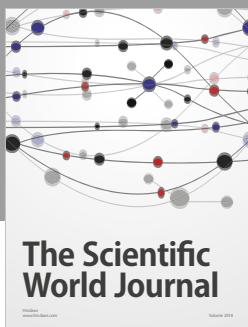
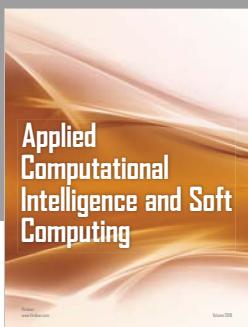
The author declares that they have no conflicts of interest.

## References

- [1] J. McGonigal, *Reality Is Broken: Why Games Make Us Better and How They Can Change The World*, Vintage, London, UK, 2012.

- [2] G. Calleja, "Digital games and escapism," *Games and Culture*, vol. 5, no. 4, pp. 335–353, 2010.
- [3] A. DeSmet, D. Thompson, T. Baranowski, A. Palmeira, M. Verloigne, and I. De Bourdeaudhuij, "Is participatory design associated with the effectiveness of serious digital games for healthy lifestyle promotion? A meta-analysis," *Journal of Medical Internet Research*, vol. 18, no. 4, 2016.
- [4] T. M. Connolly, E. A. Boyle, E. MacArthur, T. Hainey, and J. M. Boyle, "A systematic literature review of empirical evidence on computer games and serious games," *Computers & Education*, vol. 59, no. 2, pp. 661–686, 2012.
- [5] K. Lohse, N. Shirzad, A. Verster, N. Hodges, and H. F. Van der Loos, "Video Games and Rehabilitation," *Journal of Neurologic Physical Therapy*, vol. 37, no. 4, pp. 166–175, 2013.
- [6] P. Sweetser and P. Wyeth, "GameFlow," *Computers in Entertainment*, vol. 3, no. 3, 2005.
- [7] K. M. Gilledge, A. Dix, and J. Allanson, "Affective videogames and modes of affective gaming: Assist me, challenge me, emote me," in *Proceedings of the 2nd International Conference on Digital Games Research Association: Changing Views: Worlds in Play (DiGRA '05)*, 20, 16 pages, Vancouver, Canada, June 2005.
- [8] R. Koster, *A theory of fun for game design*. Sebastopol (Calif.), O'Reilly Media, 2014.
- [9] J. Chen, "Flow in games (and everything else)," *Communications of the ACM*, vol. 50, no. 4, pp. 31–34, 2007.
- [10] T. W. Malone, "Toward a theory of intrinsically motivating instruction," *Cognitive Science*, vol. 5, no. 4, pp. 333–369, 1981.
- [11] M. Csikszentmihalyi, *Flow: The Psychology of Optimal Experience*, Harper Row, New York, NY, USA, 2009.
- [12] M. M. Peeters, *Personalized Educational Games-Developing agent-supported scenario-based training [Ph.D. thesis]*, The Dutch Graduate School for Information and Knowledge Systems, 2014.
- [13] R. Hunicke and V. Chapman, "AI for Dynamic Difficulty Adjustment in Games," in *Proceedings of the Challenges in Game Artificial Intelligence AAAI Workshop*, pp. 91–96, San Jose, Calif., USA, 2004.
- [14] G. Andrade, G. Ramalho, H. Santana, and V. Corruble, "Extending reinforcement learning to provide dynamic game balancing," in *Proceedings of the Workshop on Reasoning, Representation, and Learning in Computer Games, 19th International Joint Conference on Artificial Intelligence (IJCAI)*, 12, 7 pages, Edinburgh, United Kingdom, 2005.
- [15] R. A. Bartle, *Designing Virtual Worlds*, New Riders, Berkeley, Calif., USA, 2006.
- [16] R. Hunicke, "The case for dynamic difficulty adjustment in games," in *Proceedings of the ACM SIGCHI International Conference on Advances in Computer Entertainment Technology (ACE '05)*, pp. 429–433, Valencia, Spain, June 2005.
- [17] S. Poole, *Trigger Happy Videogames and The Entertainment Revolution*, Arcade Publishing, New York, NY, USA, 2007.
- [18] S. Xue, M. Wu, J. Kolen, N. Aghdaie, and K. A. Zaman, "Dynamic Difficulty Adjustment for Maximized Engagement in Digital Games," in *Proceedings of the 26th International Conference*, pp. 465–471, Perth, Australia, April 2017.
- [19] C. V. Segundo, K. Emerson, A. Calixto, and R. P. Gusmao, "Dynamic difficulty adjustment through parameter manipulation for Space Shooter game," in *Proceedings of SB Games*, Brazil, 2016.
- [20] H. Hsieh, *Generation of Adaptive Opponents for a Predator-Prey Game*, Asia University, 2008.
- [21] S. Bunian, A. Canossa, R. Colvin, and M. S. El-Nasr, "Modeling individual differences in game behavior using HMM," in *Proceedings of the 13th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE-17)*, 2017.
- [22] M. M. Khajah, B. D. Roads, R. V. Lindsey, Y.-E. Liu, and M. C. Mozer, "Designing engaging games using Bayesian optimization," in *Proceedings of the 34th Annual Conference on Human Factors in Computing Systems, CHI 2016*, pp. 5571–5582, San Jose, Calif, USA, May 2016.
- [23] A. Hintze, R. S. Olson, and J. Lehman, "Orthogonally evolved AI to improve difficulty adjustment in video games," in *European Conference on the Applications of Evolutionary Computation*, vol. 9597 of *Lecture Notes in Computer Science*, pp. 525–540, Springer International Publishing, Cham, Switzerland, 2016.
- [24] C. Pedersen, J. Togelius, and G. N. Yannakakis, "Modeling player experience in Super Mario Bros," in *Proceedings of the 2009 IEEE Symposium on Computational Intelligence and Games (CIG)*, pp. 132–139, Milano, Italy, September 2009.
- [25] G. N. Yannakakis and J. Hallam, "Game and player feature selection for entertainment capture," in *Proceedings of the 2007 IEEE Symposium on Computational Intelligence and Games*, pp. 244–251, Honolulu, Hawaii, USA, April 2007.
- [26] N. Shaker, G. Yannakakis, and J. Togelius, "Towards automatic personalized content generation for platform games," in *Proceedings of the 6th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, AIIDE 2010*, pp. 63–68, Stanford, Calif, USA, October 2010.
- [27] C. Pedersen, J. Togelius, and G. N. Yannakakis, "Modeling player experience for content creation," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 2, no. 1, pp. 54–67, 2010.
- [28] M. Jennings-Teats, G. Smith, and N. Wardrip-Fruin, "Polymorph: A model for dynamic level generation," in *Proceedings of the 6th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, AIIDE 2010*, pp. 138–143, Stanford, Calif, USA, October 2010.
- [29] L. V. Carvalho, A. V. M. Moreira, V. V. Filho, M. Túlio, C. F. Albuquerque, and G. L. Ramalho, "A Generic Framework for Procedural Generation of Gameplay Sessions," in *Proceedings of the SB Games 2013, XII SB Games*, São Paulo, Brazil, 2013.
- [30] P. Spronck, I. Sprinkhuizen-Kuyper, and E. Postma, "Online adaptation of game opponent AI with dynamic scripting," *International Journal of Intelligent Games & Simulation*, vol. 3, no. 1, pp. 45–53, 2004.
- [31] Z. Simpson, "The In-game Economics of Ultima Online," in *Proceedings of the Game Developers Conference*, San Jose, Calif, USA, 2000.
- [32] J. Togelius, R. DeNardi, and S. M. Lucas, "Making racing fun through player modeling and track evolution," in *Proceedings of the Workshop Adaptive Approaches Optim. Player Satisfaction Comput. Phys. Games*, p. 70, 2006.
- [33] J. Hagelback and S. J. Johansson, "Measuring player experience on runtime dynamic difficulty scaling in an RTS game," in *Proceedings of the 2009 IEEE Symposium on Computational Intelligence and Games (CIG)*, pp. 46–52, Milano, Italy, September 2009.
- [34] C. H. Tan, K. C. Tan, and A. Tay, "Dynamic game difficulty scaling using adaptive behavior-based AI," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 3, no. 4, pp. 289–301, 2011.
- [35] Y. A. Sekhavat, "MPRL: Multiple-Periodic Reinforcement Learning for difficulty adjustment in rehabilitation games,"

- in *Proceedings of the 5th IEEE International Conference on Serious Games and Applications for Health, SeGAH 2017*, Perth, Australia, April 2017.
- [36] X. Li, S. He, Y. Dong et al., “To create DDA by the approach of ANN from UCT-created data,” in *Proceedings of the 2010 International Conference on Computer Application and System Modeling (ICCASM 2010)*, pp. V8-475–V8-478, Taiyuan, China, October 2010.
  - [37] J. Yang, Y. Gao, S. He et al., “To Create Intelligent Adaptive Game Opponent by Using Monte-Carlo for Tree Search,” in *Proceedings of the 2009 Fifth International Conference on Natural Computation*, pp. 603–607, Tianjian, China, August 2009.
  - [38] H. Yin, L. Luo, W. Cai, Y.-S. Ong, and J. Zhong, “A data-driven approach for online adaptation of game difficulty,” in *Proceedings of the 2015 IEEE Conference on Computational Intelligence and Games, CIG 2015*, pp. 146–153, Tainan, Taiwan, September 2015.
  - [39] A. Ebrahimi and M.-R. Akbarzadeh-T, “Dynamic difficulty adjustment in games by using an interactive self-organizing architecture,” in *Proceedings of the 2014 Iranian Conference on Intelligent Systems, ICIS 2014*, Iran, February 2014.
  - [40] M. A. Potter and K. A. de Jong, “Cooperative coevolution: an architecture for evolving coadapted subcomponents,” *Evolutionary Computation*, vol. 8, no. 1, pp. 1–29, 2000.
  - [41] H. Takagi, “Interactive evolutionary computation: fusion of the capabilities of EC optimization and human evaluation,” *Proceedings of the IEEE*, vol. 89, no. 9, pp. 1275–1296, 2001.
  - [42] J. K. Olesen, G. N. Yannakakis, and J. Hallam, “Real-time challenge balance in an RTS game using rtNEAT,” in *Proceedings of the 2008 IEEE Symposium on Computational Intelligence and Games, CIG 2008*, pp. 87–94, Perth, Australia, December 2008.
  - [43] A. Stein, Y. Yotam, R. Puzis, G. Shani, and M. Taieb-Maimon, “EEG-triggered dynamic difficulty adjustment for multiplayer games,” *Entertainment Computing*, vol. 25, pp. 14–25, 2018.
  - [44] D. Afshari, E. M. Peck, E. T. Solovey et al., “Dynamic difficulty using brain metrics of workload,” in *Proceedings of the 32nd Annual ACM Conference on Human Factors in Computing Systems, CHI 2014*, pp. 3797–3806, Toronto, Ontario, Canada, May 2014.
  - [45] H. D. B., K. Mikami, and K. Kondo, “Adaptable game experience based on player’s performance and EEG,” in *Proceedings of the 2017 Nicograph International (NicoInt)*, pp. 1–8, Kyoto, Japan, June 2017.



# Exploring Dynamic Difficulty Adjustment in Videogames

Gabriel K. Sepulveda, Felipe Besoain, and Nicolas A. Barriga

**Abstract**—Videogames are nowadays one of the biggest entertainment industries in the world. Being part of this industry means competing against lots of other companies and developers, thus, making fanbases of vital importance. They are a group of clients that constantly support your company because your video games are fun. Videogames are most entertaining when the difficulty level is a good match for the player’s skill, increasing the player engagement. However, not all players are equally proficient, so some kind of difficulty selection is required. In this paper, we will present Dynamic Difficulty Adjustment (DDA), a recently arising research topic, which aims to develop an automated difficulty selection mechanism that keeps the player engaged and properly challenged, neither bored nor overwhelmed. We will present some recent research addressing this issue, as well as an overview of how to implement it. Satisfactorily solving the DDA problem directly affects the player’s experience when playing the game, making it of high interest to any game developer, from independent ones, to 100 billion dollar businesses, because of the potential impacts in player retention and monetization.

**Keywords**—Dynamic Difficulty Adjustment, Videogames, Artificial Intelligence

## I. INTRODUCTION

WHEN someone is playing a videogame, his goal is to have fun. Game developers must infuse the games with this fun. The fun factor is composed of four axes: fantasy, curiosity, player control, and challenge. If kept in balance the player will stay entertained [1]. The challenge axis is the most difficult to control because to do so, the game difficulty and player skill must match.

Setting a single difficulty level fit for every player is not possible. A solution is to allow the player to choose a difficulty level. This method has some drawbacks, such as having a limited set of difficulty levels, creating gaps where players can fall and having difficulty progression that mismatch player learning curves. Also, by making the player aware of the change in difficulty the game experience is affected. Over the last decade, there have been multiple publications related to the improvement of this issue, using Dynamic Difficulty Adjustment (DDA) [2]. As a result, there are various algorithms the developer can use to implement DDA, and choosing the most appropriate one can be a difficult task. The lack of experience makes choosing and correctly applying the algorithm harder, leading to a poor implementation causing conflicts in the game.

G.K. Sepulveda, F. Besoain, and N.A. Barriga are with the Escuela de Ingeniería en Desarrollo de Videojuegos y Realidad Virtual, Facultad de Ingeniería, at Universidad de Talca. Campus Talca, Chile. (e-mail: gsepulveda17@alumnos.utalca.cl, fbesoain@utalca.cl, nbarriga@utalca.cl).

Corresponding Author: N.A. Barriga. (e-mail: nbarriga@utalca.cl)

In the following section we will briefly describe the Dynamic Difficulty Adjustment problem. Section III will introduce the readers to the assessment of player’s skill levels. Section IV gives an overall explanation of how a DDA system works, while section V reviews the implementations of different approaches to DDA. Finally, we close with a summary and some pointers for future work.

## II. BACKGROUND

When doing an activity that is neither boring nor frustrating, the person becomes engrossed in said activity, being able to perform longer and keep focused on the task. This state of mind is called the flow channel (flow) [3] and is present in all fields. This concept was later on introduced in the videogames area by Koster [4].

In figure 1, it’s possible to note that when the difficulty of the game is higher than the players skills the activity becomes frustrating pushing the player into a state of anxiety. In contrast when the player skills are higher than the difficulty, the game is too easy, pushing the player into a state of boredom. When neither of those happen, the user is faced by a challenge whose difficulty level matches the player’s skill, enabling him to enter the flow. Providing a series of challenges allows the player to stay in the flow for longer periods of time.

It is important to note that taking breaks between the challenges will prevent overwhelming the player. By alternating a series of constant challenges with break times it’s possible to create a game that enthralls the player and keeps him playing.

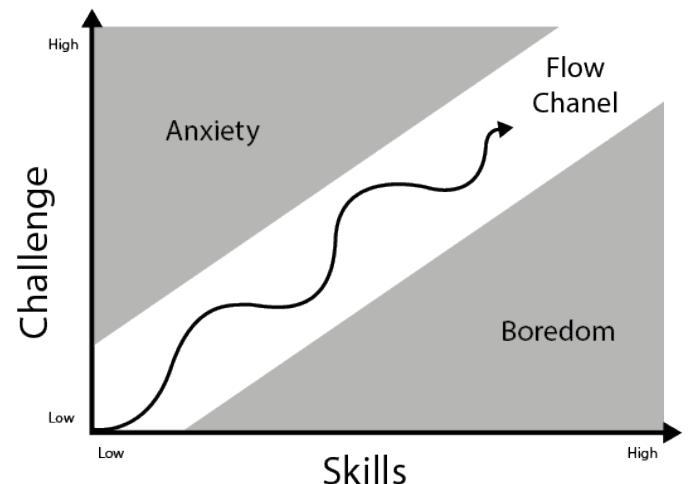


Fig. 1. Flow Channel

Hunicke proposes the creation of a system with techniques for stochastic inventory management that periodically examines the player progress and dynamically adjust the difficulty of the game challenges to adjust the player's overall experience [5]. Later on, he observes that a DDA system can improve the player experience without the need for any sophisticated AI [6].

DDA is an AI-based system that allows the change of attributes and behaviors within the game in runtime. DDA measure the player performance and change the game difficulty to match the player skills. As a result it creates the challenge used to guide the player into the flow zone. A good DDA must be able to track the player skill level and adapt to it. Changes in difficulty must follow the player learning curve and go unnoticed by the player [2].

A DDA system that fulfills these requirements can increase the player confidence in his chances to beat the game when presented with a hard challenge [7] and the core engagement of the player, resulting in an increased gameplay duration [8].

### III. ASSESSING PLAYER SKILLS

The first step needed to implement a DDA system is an evaluation of the player's performance. Through a set of predefined variables it's possible to asses if the difficulty is fit for the player. This is done by comparing the current in-game value of this variables with their expected value.

#### A. Variables

In order to choose the variables used to evaluate the player performance it is important to have a clear notion of what is considered as a failure, a success or a skill that the player needs to develop within the game. Usually, they'll correspond to game rules and win or loss conditions [9].

A good example is given in a study using Pac-Man as a test-bed. By measuring the number of hits on the maze walls, the number of keys pressed and the number of direction switches, it is possible to have an idea of what the player skills are [10]. Also the lives lost and pills collected versus time are good indicators of failure and success rate.

Another example is given by the analysis performed to the Multiplayer Online Battle Arena (MOBA) game Defense of the Ancient (DotA)<sup>1</sup> In this case the level reached versus time, and towers destroyed versus time, are used to measure success, and deaths versus time, used for failure [11]. Additionally, we can count the number of minions killed, heroes killed, items completed and missed abilities, among others, to check the player skill level.

As in the previous examples, all games can include variables that indicate the current state of the player and give information about his performance, success and failure ratio, and his learning process. The variables selected will depend on the specific game in which the DDA system is implemented.

Note that a high number of variables will result in a more precise difficulty adjustment but will also consume more memory, in the other hand a low number of variables will

result in poor adjustment. The amount of variables chosen will depend on the complexity of the game but for most of the studies a number of three to five variables is enough.

#### B. Data Collection

For the DDA system to keep track of the chosen variables, the game has to overwrite their value. The tracking can be event-triggered or permanent.

Event-triggered tracking is for variables that change when fulfilling a condition or performing an action. In this case, the method in charge of triggering the event can call the DDA system class and change the variable value.

An example of this are the variables used in the previously mentioned Pac-Man research [10]. The number of hits on maze walls, number of keys pressed or number of direction changes must be actualized just when the corresponding action happens.

On the other hand, permanent tracking is applied to variables that are always in game. The game has to call the DDA system class and update the values of the variables in the game loop. Setting a minimum time between updates is recommended to reduce processing.

Examples of variables that need permanent tracking are player health, player gold, and game progress.

#### C. Reference Point

With the collected data, it's possible to assess the player skills using the performance of a player that matches the game difficulty as a reference point. The chances of finding a player that perfectly matches the game difficulty are zero. Thus, we need to find a method that can provide the reference point.

Setting the reference point based on the beliefs of what it should be is the worst way to do so. Even with years of experience, it's unlikely to guess the correct value.

One option is to use an AI agent capable of playing the game. The data of the AI play-through will be then used as a reference point. In order to get reliable information the agent must play the game many times, a bigger number of iterations gives more precise information. An agent that plays the game perfectly isn't useful as it has to imitate a real player, to that end an agent that can play at a medium level is required instead.

However, the best option would be to have real player data. Using the same system to get data the game can be tested with real players. Using a survey after the testing session, the data of players that had fun while playing can be used as a sample.

#### D. Data Analysis

Having set the variables to analyze, the reference point for each, and being able to save the data from the player play-through, the system has the essentials needed for the evaluation of player performance. These evaluations should be made through the use of methods that return a success, failure or skill ratio. Just subtracting the number of player deaths from the expected player deaths returns a non-representative number in most cases. The evaluation must also happen constantly in

<sup>1</sup>[https://en.wikipedia.org/wiki/Defense\\_of\\_the\\_Ancients](https://en.wikipedia.org/wiki/Defense_of_the_Ancients)

the game, for the difficulty to change at the right times. To this end, the evaluation of the player performance must happen once each X number of ticks. A low X value makes the game more adaptable, but increase the process cost of the game and might not be needed, selecting the right X value will depend on the game genre.

Evaluation of player performance can be done by comparing the player's current stats with the ideal or reference ones versus a delta time [10] (equation 1).

$$\text{Difficulty} = (N - Z)/D; \quad (1)$$

$$\text{Ease} = 1 - \text{Difficulty}; \quad (2)$$

Let's take, for example, a situation where the player performs an action  $N$  times in a  $D$  period of time or ticks, with the ideal or reference value being  $Z$ . Using a value set that abides by  $0 \leq (N - Z) \leq D$  will return a normalized value, allowing an easier interpretation of the results.

As the enjoyment of a game is greater when the game is equally hard and easy, the point where these two values merge is the desired game state. A return value close to 0.5 fulfills this condition. The chances of getting the desired value are low, that's why leaving a proper margin is recommended. The margin must be defined by the game developer based on the situation. Leaving a 0.1 error margin would leave us with a range between [0.4, 0.6], where the game difficulty is acceptable.

This method also allows calculating the player global proficiency by calculating the average of all variables or a weighting of them. Consider that it is convenient for the global proficiency, or player global performance, to be a normalized value. To this end, the sum of all weights must also be normalized.

Another method is to assess the player current performance ( $C_p$ ) versus the expected player performance at that time ( $E_p[t]$ ), which would be our reference point.

$$\text{Performance} = C_p/E_p[t]; \quad (3)$$

In this manner, a value next to 1 means that the challenge is appropriate for the player. Same as in the previous method a error margin defined by the game developer is needed as getting the exact value is unlikely. For example, setting the error margin of 0.2 give us a range [0.8, 1.2]. This mean that values lower than 0.8 indicates the game is too hard and values greater than 1.2 means the game is too easy.

This method also allows to create a ranking of the player global performance by adding the results. Both methods can be modified to better suit the game developer preferences and needs.

#### IV. IMPLEMENTATION

There are several different DDA methods proposed in the literature. We will focus on the more straightforward ones, with the purpose of giving the reader some insight on how to implement the DDA system into his projects. For more information, a recent review of DDA research from 2009 to

TABLE I  
DDA APPROACHES

| Author(s)                          | Year | Approach   |
|------------------------------------|------|--|
| Hunicke and Chapman                | 2004 | Hamlet System [5]  |
| Spronck et al.                     | 2006 | Dynamic Scripting [12]                                     |
| Pedersen, Togelius, and Yannakakis | 2009 | Single and multi-layered perceptrons [13]                  |
| Hagelback and Johansson            | 2009 | Reinforcement Learning [14]<br>Upper Confidence Bound      |
| Li et al.                          | 2010 | for Trees and Artificial Neural Networks [15]              |
| Ebrahimi and Akbarzadeh-T          | 2014 | Self-organizing System and Artificial Neural Networks [10] |
| Sutoyo et al.                      | 2015 | Metrics [16]   |
| Xue et al.                         | 2017 | Probabilistic Methods [8]                                  |
| Stein et al.                       | 2018 | EEG-triggered dynamic difficulty adjustment [17]           |

2018 is available [2]. Table I summarizes the approaches found in the literature.

At the start of the game, there is no data of the player performance, and so, the developer must set the difficulty based on the average testing results. Then, the game must quickly adapt to the player performance. Playable tutorials are a good opportunity to get early information on the player performance without having to make him go through any real challenge. Afterward, the change of difficulty should happen less often and be smaller, allowing for the growth of the player skills.

The biggest challenge while changing the difficulty of the game in real time is avoiding the player noticing the change. To avoid this, performing the difficulty change at times where the player is not aware is a must. Such cases are times when the player is dead, change of scenes, features or elements the player has not yet seen. If the change is subtle enough, elements that the player is not currently seeing, non-perceptible changes as most of the element specific variable values changes, or minor behavior changes can be performed without fearing the player to notice it, as long as the changes don't happen too often, and aren't too extreme. There must be an upper and lower limit for how much a variable can change, as well as a time threshold for the changes to happen and a maximum number of changes for each update and stage. The changes to be performed can be added to a queue as functors or callbacks to be executed at a given time. Each change can have a tag that identifies the change. When a change is going to enter the queue any changes performed with the same tag will be removed, preventing two changes to affect the same element, as this is unneeded and can cause problems.

Big changes, such as the size of a room, conspicuous change of behavior with no reason in the gameplay, and others, must be executed in load screens, while changing scenes, or in sections not seen by the player.

These can also be added to the functors or callbacks queue and be executed when needed. On the other hand, changes to be performed on zones unseen by the player, but on the same

scene, can be performed immediately to prevent the player from getting to the zone with the changes undone.

The creation of mathematical functions that tells us how much the variables should be changed is probably the best way to perform these changes. The functor can receive as parameter the proficiency of the player in the corresponding field and calculate how much each variable must change to fit the player. The definition of the mathematical functions must be done with the data collected from the multiple testing phases. Creating a game with preset difficulties and make a study where players test all the difficulty levels is advisable.

Each one of the variables selected as an indicator of the player performance is directly affected by at least one factor in the game. For example, the death ratio of the player depends on the damage dealt by the enemies, the chances to evade an attack, and the aggressiveness of the enemies, among others. Linking each evaluation variable to a factor is crucial, as these factors are what is going to be changed in order to adapt the game difficulty to the player. These factors can be categorized in three sections, attributes, behaviors and events.

#### A. Attributes

Changing the value of attributes in the game is the first and most intuitive of the modifications to be done, and also, the easiest of them. Even so, the number of attributes that can create the difficulty trait in one of these variables can be immense. Specific information, like the given by reports of event-triggered parameters, can allow discerning how to properly balance the game. By comparing the player stats with the enemy who killed him, and the time of the engagement, it is possible to know if the player died because of the difference in damage, speed, range, attack ratio or others. There are also situations where the player died because his health was low before the engagement, which means that it's not the current enemy the reason for his death, but a previous one or the sum of them. The tracking of spikes in permanent game values allows a better adjustment. If there was a spike drop in health, knowing whether it was caused by a single enemy or by a group attack, can trigger the adjustment of the single enemy's stats, or the reduction in the number of enemies. If there was no spike then maybe the speed at which the waves of enemies reach the player is too fast and slowing down is required. As shown in the previous example, the traits of the characters in the game are not the only thing that can be changed, but also the number of enemies on the same zone, or even subtle things, such as the auto-aim range, the time limit for a quick time event, the dimension of a room, or the pace of the game.

#### B. Behaviors

Traits in a game can be presented not only as attributes but also as behaviors. Behaviors are the main component of complexity in a game and are not exclusive to NPCs (Non-Playable Characters), in three-in-line games the pieces have the behavior to self destruct if there are another two equal pieces at a two tiles distance (vertical or horizontal), in the same axis (x,y). Modern three-in-line have added new blocks and behaviors. These behaviors cannot be changed, as they

are the foundation of the game, but not all games have these restrictions.

In a stealth game, the watch tower can move the light following different patterns, alternating patterns, or just randomly. The more predictable the pattern is, the easier for the player. The ability to communicate with other watch towers and to detect footprints or noises are other behaviors that can be enabled, disabled, or modified to change the difficulty of the game. In a MOBA, the tower has the behavior to attack the enemies, by changing the target priority, the game changes the difficulty drastically.

#### C. Events

Events provide an alternative that doesn't require too many modifications to the existing game codebase, as does the change in attributes and behaviors, but their implementation requires more design work than the previous two, and are easier to be spotted by the player if the implementation is poor. Events are predefined occurrences that arise under certain circumstances. For example, if the player is low on health, and his performance is low, the next enemy will always drop a health potion. Conversely, if the player's health is full, and his proficiency level is high, the next enemy hit will always deal critical damage.

### V. MODELS

In this section, we will introduce a small selection of existing approaches for DDA.

#### A. Metrics

As mentioned before, it's important to identify the factors that directly influence the player performance. In the simpler use of metrics, a multiplier is applied to the variables that controls said factors. The initial value for the multipliers is to be defined by the game developer, it's recommended to use a neutral multiplicative at the start of the game as there shouldn't be any change on the factors yet. For each relationship between the variables used to evaluate player performance ( $EV$ ) and the attributes that affect the player performance ( $FV$ ), a weight is defined. Note that the maximum number of weights needed is of  $EV \times FV$ , in case that each attribute affects every variable, which means adding variables or attributes would greatly increment the number of weights to define. If possible, each variables should be affected by a few attributes, and not all of them in order to reduce the number of weights.

These weights are added to the multiplier in function of the difference between player performance ( $PP$ ) and game difficulty ( $GD$ ). The evaluation can be performed through the use of thresholds [16] or multiplying the weight with said difference  $\frac{PP}{GD}$ .

#### B. Probabilistic Methods

Probabilistic methods focus on predicting events on the game through the use of probabilistic calculations and using the probabilities in a challenge function [8].

The probabilistic calculations are used to get the expected value of factors that directly affects the player performance and act accordingly before the player faces the challenge.

As an example, in case the player is reaching a zone with 40% of his health, the probabilistic calculation will get the expected value of the total damage the player is going to suffer. This value will be returned to the challenge function that is going to evaluate whether the challenge is too difficult or too easy for the player and act accordingly before the player enters the zone.

Experiments show that the probabilistic method is effective in games organized in stages [18] or levels [8], as it allows the AI to know which calculations it has to make based on the player current location and direction or to precalculate the values of each stage.

### C. Dynamic Scripting

Dynamic Scripting is an online machine learning technique focused in the modification of behaviors of agents in the game. Dynamic Scripts (DS) are built from a set of rulebases, one for each type of agent to be modified. Each time an agent is created, the associated rulebase is used to create a new script that controls its behavior. Each rule of the rulebase has an associated weight that determine the chances of selecting the rule. The weights are adjusted based on a fitness function that evaluates the system's performance [12].

Image 2 shows an example of a Dynamic Scripting system in action. The character has an action rule-base. Variations on the same rules are performed to give the algorithm more options. The DS selects rules from the rule base to create the behavior script.

In order to have a good performance, a Dynamic Scripting implementation has to meet certain requirements [12], [19]:

**Speed:** Algorithms in online machine learning must be computationally fast since they take place during gameplay.

**Effectiveness:** The created scripts should be at least as challenging as manually designed ones.

**Robustness:** The learning mechanism must be able to cope with a significant amount of randomness inherent in most commercial gaming mechanisms.

**Efficiency:** The learning process should rely on a small number of trials, since a player experiences a limited number of encounters with similar groups of opponents.

**Clarity:** Adaptive game AI must produce easily interpretable results, because game developers distrust learning techniques of which the results are hard to understand.

**Variety:** Adaptive game AI must produce a variety of different behaviors, because agents that exhibit predictable behavior are less entertaining than agents that exhibit unpredictable behavior.

**Consistency:** The average number of learning opportunities needed for adaptive game AI to produce successful results should have a high consistency to ensure that their achievement is independent both from the behavior of the human player, and from random fluctuations in the learning process.

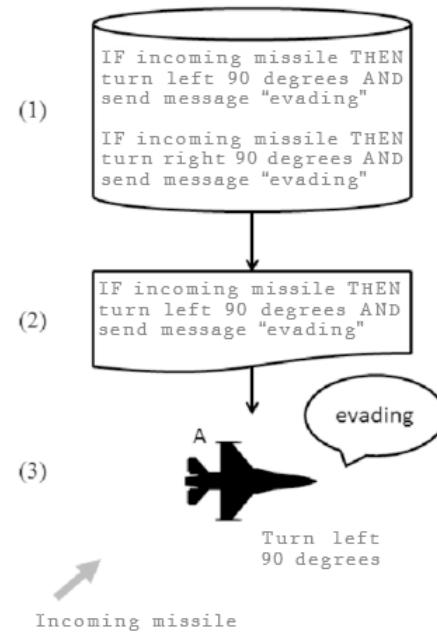


Fig. 2. Dynamic Scripting [20]

**Scalability:** Adaptive game AI must be able to scale the difficulty level of its results to the skill level of the human player.

As Dynamic Scripting is a continually learning AI, it might keep improving until it reaches a point where it can always defeat the player. In DDA the objective isn't to beat the player but to provide a fitting challenge, so in order to avoid the AI surpassing the player skills some modifications have been added to traditional Dynamic Scripting.

**Weight Clipping** is a technique where a maximum  $W$  value is determined, preventing the weights to grow over  $W$  [21]. Normally, a set of rules with a large win rate will result in an AI with high performance, thus increasing the weights of those rules and increasing the chances of being selected, resulting on an AI that keeps defeating the player. By setting a low  $W$  value, the chances of the rule set being selected are reduced, allowing the AI to pick tactics that make him lose against the player.

Top Culling technique, as Weight Clipping, sets a maximum  $W$  value and allows the weights to grow limitless but weights with values that surpasses  $W$  will not be selected to generate scripts [21].

Another technique used to increase the player enjoyment is the **Adrenaline Rush**. This technique is based on the assumption that the player's learning rate is high at the start of the game but drops through it. A learning limit is set to restrict the weights adjustment, and a maximum player fitness delta  $P$  is defined. By constantly measuring the player fitness and keeping track of the previous value a player fitness difference between the previous and current state can be calculated. When this player fitness delta drops below  $P$  the learning limit of the weights is decreased so the AI doesn't overgrow the player [22].

Finally, it's possible for the Dynamic Scripting's fitness function to minimize the difference between it's performance and the player's, rather than to maximize absolute performance. This way the AI will adjust the weights to select rules that increase the chances of the game difficulty fitting the player skills, instead of the rules that make it more likely to win.

## VI. CONCLUSION

Dynamic Difficulty Adjustment (DDA) is a technique that allows the developer to give the player a game that adapts itself to fit him, thus increasing player engagement. In recent years the amount of research in DDA has increased. As a result, there are many approaches to implement it, with the main difference being the method used to change the attributes and behaviors of the game. Because of that, it is possible to have a general structure of how to implement the DDA. The proposed structure is separated into two main parts, measuring the player proficiency and adjusting the game accordingly, using the method preferred by the developer.

Implementing a Dynamic Difficulty Adjustment system is a powerful tool that allows for a better game experience. Its implementation is based mainly in data collection from test subjects and the player playing the game, and use of metric and mathematical functions to define the changes to be performed. By adjusting variables and behaviors within the game create challenges fit to the player, but careful planning is needed. Poor implementation can result in having the opposite effect and overuse can cause high processing consumption from the game.

### A. Future Work

DDA is still a new field of research and much can be done to increase its effect in player engagement. New methods for implementation are always needed but research on optimization of already existing ones or creation of tools that enable an easier DDA implementation are also interesting.

Research can be done to reduce the process required for the evaluation of the fitness function, in order to be able to add more variables and create even more precise DDA systems. As well as the creation of a method that enable a low cost adjustment of weights allowing the developer to manipulate more factors of the game.

Integration of Behaviors Trees and Finite-State Machines to the Dynamic Scripting approach, where all possible three nodes or states and transitions are preprogrammed, and the used ones are selected by the DDA system is another promising research area.

Finally the creation of tools that allows easy DDA implementation for all game styles and that can be used in the most popular frameworks or engines for game development, such as Unity or Unreal, is a field that is still untouched.

## REFERENCES

- [1] T. W. Malone, "Toward a theory of intrinsically motivating instruction," *Cognitive Science*, vol. 5, no. 4, pp. 333–369, 1981.
- [2] Z. Mohammad, "Dynamic difficulty adjustment (DDA) in computer games: A review," *Advances in Human-Computer Interaction*, vol. 2018, no. 12, pp. 333–369, 2018.
- [3] M. Csikszentmihalyi, *Flow: The Psychology of Optimal Experience*. New York : Harper and Row, 2009.
- [4] R. Koster, *Theory of fun for game design*. O'Reilly Media, Inc., 2013.
- [5] R. Hunicke and V. Chapman, "AI for dynamic difficulty adjustment in games, challenges in game artificial intelligence aaaiworkshop," 2004.
- [6] R. Hunicke, "The case for dynamic difficulty adjustment in games," in *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology*. ACM, 2005, pp. 429–433.
- [7] T. Constant and G. Levieux, "Dynamic difficulty adjustment impact on players' confidence," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, ser. CHI '19. New York, NY, USA: ACM, 2019, pp. 463:1–463:12. [Online]. Available: <http://doi.acm.org.utalca.idm.oclc.org/10.1145/3290605.3300693>
- [8] S. Xue, M. Wu, J. Kolen, N. Aghdaie, and K. A. Zaman, "Dynamic difficulty adjustment for maximized engagement in digital games," in *Proceedings of the 26th International Conference on World Wide Web Companion*. International World Wide Web Conferences Steering Committee, 2017, pp. 465–471.
- [9] M. Hendrix, T. Bellamy-Wood, S. McKay, V. Bloom, and I. Dunwell, "Implementing adaptive game difficulty balancing in serious games," *IEEE Transactions on Games*, pp. 1–1, 2018.
- [10] A. Ebrahimi and M. Akbarzadeh-T, "Dynamic difficulty adjustment in games by using an interactive self-organizing architecture," in *2014 Iranian Conference on Intelligent Systems (ICIS)*, Feb 2014, pp. 1–6.
- [11] M. P. Silva, V. d. N. Silva, and L. Chaimowicz, "Dynamic difficulty adjustment through an adaptive AI," in *2015 14th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*, Nov 2015, pp. 173–182.
- [12] P. Spronck, M. Ponsen, I. Sprinkhuizen-Kuyper, and E. Postma, "Adaptive game AI with dynamic scripting," *Machine Learning*, vol. 63, no. 3, pp. 217–248, 2006.
- [13] C. Pedersen, J. Togelius, and G. N. Yannakakis, "Modeling player experience in super mario bros," in *2009 IEEE Symposium on Computational Intelligence and Games*, Sep. 2009, pp. 132–139.
- [14] J. Hagelback and S. J. Johansson, "Measuring player experience on runtime dynamic difficulty scaling in an rts game," in *2009 IEEE Symposium on Computational Intelligence and Games*, Sep. 2009, pp. 46–52.
- [15] Xinyu Li, Suojia He, Yue Dong, Qing Liu, Xiao Liu, Yiwen Fu, Zhiyuan Shi, and Wan Huang, "To create DDA by the approach of ANN from UCT-created data," in *2010 International Conference on Computer Application and System Modeling (ICCASM 2010)*, vol. 8, Oct 2010, pp. V8–475–V8–478.
- [16] R. Sutoyo, D. Winata, K. Oliviani, and D. M. Supriyadi, "Dynamic difficulty adjustment in tower defence," *Procedia Computer Science*, vol. 59, pp. 435–444, 2015.
- [17] A. Stein, Y. Yotam, R. Puzis, G. Shani, and M. Taieb-Maimon, "EEG-triggered dynamic difficulty adjustment for multiplayer games," *Entertainment computing*, vol. 25, pp. 14–25, 2018.
- [18] C. Segundo, K. Emerson, A. Calixto, and R. Gusmao, "Dynamic difficulty adjustment through parameter manipulation for Space Shooter game," in *Proceedings of SB Games*, 2016.
- [19] P. Spronck, I. Sprinkhuizen-Kuyper, and E. Postma, "Online adaptation of game opponent ai in simulation and in practice," in *Proceedings of the 4th International Conference on Intelligent Games and Simulation*, 2003, pp. 93–100.
- [20] A. Toubman, J. J. Roessingh, P. Spronck, A. Plaat, and J. van den Herik, "Improving air-to-air combat behavior through transparent machine learning," in *Proceedings of the IITSEC 2014 Conference*, 2014.
- [21] P. Spronck, I. Sprinkhuizen-Kuyper, and E. Postma, "Enhancing the performance of dynamic scripting in computer games," in *International Conference on Entertainment Computing*. Springer, 2004, pp. 296–307.
- [22] J. J. P. Arulraj, "Adaptive agent generation using machine learning for dynamic difficulty adjustment," in *2010 International Conference on Computer and Communication Technology (ICCCT)*, Sep. 2010, pp. 746–751.

# AI for Game Production

Mark Owen Riedl

School of Interactive Computing  
Georgia Institute of Technology  
riedl@cc.gatech.edu

Alexander Zook

School of Interactive Computing  
Georgia Institute of Technology  
a.zook@gatech.edu

**Abstract**—A number of changes are occurring in the field of computer game development: persistent online games, digital distribution platforms and portals, social and mobile games, and the emergence of new business models have pushed game development to put heavier emphasis on the live operation of games. Artificial intelligence has long been an important part of game development practices. The forces of change in the industry present an opportunity for Game AI to have new and profound impact on game production practices. Specifically, Game AI agents should act as “producers” responsible for managing a long-running set of live games, their player communities, and real-world context. We characterize a confluence of four major forces at play in the games industry today, together producing a wealth of data that opens unique research opportunities and challenges for Game AI in game production. We enumerate 12 new research areas spawned by these forces and steps toward how they can be addressed by data-driven Game AI Producers.

## I. INTRODUCTION

Over the past several years the game industry has undergone a series of major changes. The increasing prominence of persistent online games, digital distribution platforms and portals, mobile and social games, and the emergence of new business models have all changed fundamental aspects of making and playing games. Developers increasingly focus on the live operation of a game, rather than creating a boxed and finalized product. Players are more diverse, have access to games in more places and at more times, and produce more data and content for developers to leverage than ever before. Across these changes four forces have come to the fore:

- 1) Games and cross-game play is increasingly persistent and presenting longer-term experiences
- 2) Game developers are starting to see their game titles as an ecosystem wherein players may move from game to game
- 3) Player communities and social gameplay have gained prominence
- 4) Developers and players have a growing interest in coupling the real world and virtual world(s)

Some of these forces have been around for a number of years, while others are just beginning to emerge. The consequence of these forces is a profound opportunity for Artificial Intelligence (AI), Computational Intelligence (CI), and Machine Learning (ML) in games (collectively referred to as *Game AI*) to play an even greater role in the development of computer games and the delivery of engaging real-time experiences to players.

Through these forces we see an opportunity for Game AI to address new research questions that have the potential to dramatically impact game development practices. The most

significant consequence of the shifting landscape of computer game development is the generation of massive amounts of data. Researchers and game developers can leverage this data in a new paradigm of data-driven Game AI focused on how to use these sources of data to improve game development practices and to deliver more engaging experiences. However, we are not merely advocating that researchers and game developers adopt data-driven analogues to existing Game AI practices (although that would be a worthy endeavor). Instead, we are proposing that the market forces enumerated above are forcing the industry to change how they think about game development processes. The modern development environment presents significant challenges to scalability—of both the practice of creating games and also the size and scope of games themselves—that are readily addressed by artificial intelligent, computational intelligence, and machine learning research.

Game AI was initially about supporting the interaction between player and the game itself. Recently there has been a push for procedural content generation as a means to support and augment the game developer on a game by game basis. In this paper, we present our desiderata on the future of Game AI in which intelligent systems support the entire game production pipeline from game creation to live operation, and across a number of games and game genres. This perspective of Game AI for production does not supplant or replace prior perspectives on Game AI, but presents a new lens through which to see an expanded role for Game AI in computer game production. We will enumerate 12 novel research questions that arise when considering the role of AI as Producer and discuss steps toward how these challenges may be addressed.

## II. THE SHIFTING LANDSCAPE OF GAME DEVELOPMENT

In this section, we enumerate some of the prominent new forces that are shaping the way that computer game development companies create games. Many of these forces have arisen as a confluence of advances in computing technology and market forces that result from a maturation of the computer game industry.

*1) Persistent Games:* The rise of Massively Multiplayer Online Games (MMOGs), social games, online game portals and long-term or recurring game experiences is creating long-term data on players within games. Many games have players join, play over long periods of time, and potentially rejoin at later times. Developers are increasingly pressed to develop content that provides long-term engagement with a game, rather than a closed experience with clear beginning and end.

*2) Ecosystem of Games:* Developers have a wider variety of tools to build games and lowered barriers to distributing

games to new users through digital platforms and online portals. Together this puts greater emphasis on keeping players engaged within an ecosystem of games from a single developer, rather than focusing on experiences only within one game. Developers are driven to provide new content of multiple kinds (including meta-game objectives) that engage players across games. The growing diversity of game players has led to additional emphasis on ensuring an ecosystem of games provides a diverse range of experiences.

*3) Player Communities:* Player communities have emerged as a major driving force for the success (and failure) of games. Players generate a mass of content from reviews and walkthroughs through add-ons and full game modifications. Continually engaging communities, supporting player socializing within the community, managing how players impact one another's experiences (positively and negatively), and leveraging user-generated content are growing concerns.

*4) Coupling of Real and Virtual Worlds:* Games are more widely adopting ways to connect to the real world. Sensing systems from the Kinect and Wiimote to mobile phone GPS provide more data on players' real-world environment. Output modalities from second screen experiences and mobile phone augmented reality to virtual reality are emerging as new ways to view game content. At the same time these technologies have introduced a host of challenges around how games can interface with and use this real-world context and respond to more unstructured types of information.

Each of these forces presents new opportunities to solve problems of real-world applicability to game developers. A side effect of each of these forces is the massive amounts of data being generated about game players. While it is not necessarily the case that new research problems will require data-driven AI, CI, and ML techniques, we see this data as a tool for tackling real-world game development problems.

### III. BACKGROUND: THREE ROLES OF GAME AI

In 2001, Laird and van Lent [1] put forth their seminal argument for AI in computer games as an academic pursuit. They specifically argued that the pursuit of “human-level” AI systems could use computer games as testbeds for research because games were intermediate environments between the toy domains researchers had been using and the full complexity of the real world. While this opened Game AI as an academic endeavor, the automation of various aspects of computer games has been a part of the industrial practice of creating commercial computer games since the beginning.

*Game AI* has come to refer to the set of tools—algorithms and representations—developed specifically to aid the creation and management of interactive, real-time, digital entertainment experiences. While games are played by humans, there are a number of aspects of the game playing experience that must be automated: roles that would be best performed by humans but are not practical to do so:

- Opponents and enemies that are meant to survive for only a short time before losing.
- Non-player characters in roles that are not “fun” to play such as shopkeepers, farmers, or victims.

- Companions in single-player experiences and non-player characters in support roles.
- Drama management at scale.
- Game designer for personalized experiences at scale.

As we go down this list, Game AI is charged with taking progressively more responsibility for the quality of the human player's experience in the game.

We group Game AI approaches into three broad roles that AI, CI, or ML takes in the creation and delivery of engaging entertainment experiences. Each role targets a different stakeholder: players, designers, and producers. The first role is *AI as Actor*, in which the Game AI system mediates between the player and the game to create a compelling real time experience. The most common manifestation of AI under this paradigm is controlling or managing bots and non-player characters. The second role is *AI as Designer*, in which the AI mediates between a game designer and a single player-game system. Under this paradigm, AI systems might procedurally generate game content or adapt the game to particular players to scale the game development process. Finally, we propose a third role, *AI as Producer*, in which the AI mediates between game producers and a number of systems of players, designers, and games. Figure 1 shows how the different roles relate to each other and the three primary human stakeholders.

These three roles are not distinct phases in the pursuit of Game AI, but overlapping sets of concerns and driving problems, all of which need to be pursued individually or in unison. We see AI Producers as a superset of AI Designers, encompassing a broader set of research questions. Equivalently, we see this as a shift from Game AI for game design to Game AI for game production. In industry the differences between designers and producers have blurred as technical barriers to content production have lowered and gameplay and business (e.g. monetization and marketing) are more tightly coupled.

#### A. Artificial Intelligence as Actor

Historically, the earliest uses of artificial intelligence in computer games was to mediate between users and the game. AI served the role of an artificial human opponent or playmate, enabling play without requiring other people or filling roles humans would be loathe to fill in a game. Compared to non-Game AI, the intelligence built into games places a greater emphasis on creating engaging and entertaining experiences for users, rather than maximizing a utility function such as score or win/loss rates.

For the AI as Actor role, research has focused on non-player character (NPC) path planning and decision making. Game agent decision making emphasizes the believability of characters to support the suspension of disbelief that the player is interacting with software instead of a monster, human opponent, or human companion. These problems are still being pursued by industry developers and academic researchers.

*Drama management* is another way for AI to mediate between users and the NPCs and other aspects of a game or virtual world [2], [3], [4]. A drama manager is an omniscient agent responsible for delivering an enjoyable and coherent narrative experience to players, much as a “Dungeon Master” does the same for tabletop role playing games.

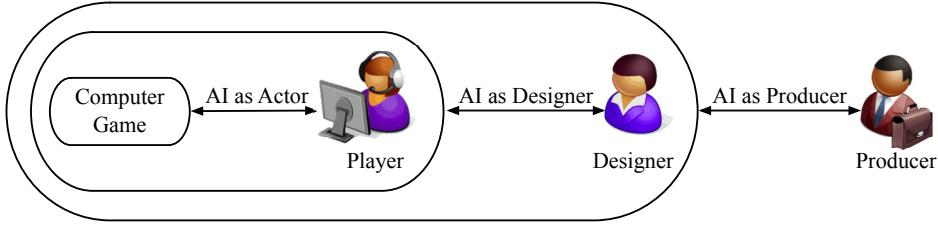


Fig. 1. Three roles of Game AI and their human stakeholders. *AI as Actor* describes how AI mediates between the player and the game itself. *AI as Designer* describes how AI mediates between a single game designer and the human-computer system of player and game. *AI as Producer* is a new role describing how AI mediates between producers and many systems of designers, users, and games.

### B. Artificial Intelligence as Designer

The second role of Game AI is to mediate between the human designer (or developer) and the human-computer system comprised of game and player. In our metaphor, game designers are responsible for building and defining a game, analyzing how players interact with the game, and iteratively refining a game to achieve a design vision. This paradigm for artificial intelligence is often referred to as *Procedural Content Generation* (PCG)—algorithms and representations for generating any and all components of games [5], [6], [7]. Offline content generation emphasizes producing content designers may curate or refine as a means of increasing the efficiency of the game development process. Online, or just-in-time, generation focuses on providing larger amounts of variation than human designers can achieve alone and/or tailoring content to a given user. A primary concern for PCG researchers has been (a) ways to appropriately represent game content to suit generation algorithms, while (b) providing means for users to interact with generation systems to author desired content and outcomes.

*Game adaptation* combines content generation and player modeling to enable AI designers to tailor games to individual players, emphasizing a closed loop of modeling player actions and automated adjustment of game content based on design goals for player behavior [8], player skill acquisition [9], or maximizing player enjoyment [10], [11], [12]. Game adaptation taken to its extreme introduces the problem of *game generation*—the automatic creation of entire games driven by (real or simulated) player feedback.

The availability of large data sets is having an impact on those who see AI as designer. In particular, *game analytics* involves capturing, aggregating, understanding, and visualizing player behavior to support designer understanding [13]. Player modeling research examines methods to describe and predict player behavior, potentially to be used by designers or automated AI systems in response [14]. While not a traditional domain of Game AI, player modeling has become increasingly prominent as AI agents shift to learning to adapt to players. Machine learning (e.g. [15], [16]) and evolutionary computing [17] are the primary areas currently being employed to perform these modeling tasks.

### C. Artificial Intelligence as Producer

The third role of Game AI uses a metaphor of AI as game producer. In our metaphor, producers concern themselves with the entire set of games and game content being made by a company, along with related aspects of managing

player communities. AI Producers mandate a shift from single player experiences within a closed game to long-term player experiences within an open game, understanding a player across multiple games in an ecosystem, and understanding how multiple players interact as an in-game and out-of-game community. AI Producers extend many methods of AI designers, driving a shift to model and adapt games that distinguishes characters (in-game avatars or personas) from players (agents manipulating those characters).

AI Producers also leverage a broader sense of context for enhancing game experiences including the community of players and the real-world context of their activities. AI for game production puts a premium on broadening the scope of Game AI to better integrate the increasingly pervasive nature of games into how games entertain and engage users. Games no longer are sharply bound by a single delivered product and Game AI ought to respond by incorporating these newly opened borders for new research domains. Effectively responding to these challenges will require new methods to leverage the masses of data being produced by players to improve interactive experiences.

## IV. GAME AI PRODUCER: RESEARCH QUESTIONS

AI producers will use the wealth of data being generated to meet the needs imposed by the four forces of (1) persistent games, (2) game ecosystems, (3) player communities, and (4) real-virtual world coupling. In each case these forces provide unique opportunities for data-driven approaches to Game AI that enhance player experience through richer sources of information and emerging modes of game-related interaction. Rather than replace earlier roles and stakeholders of Game AI, we assert that the new role of AI as Producer must emerge to address the novel concerns raised by the four forces. This new role for Game AI enhances and extends the AI techniques defined from earlier roles, leading to new research questions and techniques that potentially overlap multiple roles. In the next sections, we revisit the forces and the research questions that accompany them. Due to the overlapping nature of roles, in many cases research questions may address more than one role simultaneously.

### A. Persistent Games

Persistent games shift from closed games comprising time investments typically on the order of days to ongoing and extended game experiences spanning months or years generating long-term data on player history and activities. AI Producers address the full lifetime of a player, spanning the standard

business concerns of acquiring new players, retaining players over a long period of time, and reacquiring lapsed players. Key research questions around using long-term data to improve player engagement focus on: lifelong agents, gameplay support, and engagement-oriented content generation. Unifying these is a view of Game AI providing for long-term player engagement across an increasingly diverse group of players.

*1) How can an AI agent interact with players over very long periods of time?* An agent that persists for months, years, or even a user's lifetime is referred to as a *lifelong agent*. In the context of computer games, lifelong agents are NPCs that learn about you as a player over time. Lifelong agents serve as long term companions (or adversaries) that recognize and adapt to changes in players over time and use historical interactions with players to shift their behavior. That is, lifelong agents get to know players, and become familiar with the player, and familiar to the player.

Lifelong agents are relatively unexplored in the domain of games. In the domain of virtual agents for healthcare, Bickmore and colleagues have been designing and developing agents that interface with humans longitudinally [18]. In the context of games, lifelong agents may foster player empathy for companions—and enmity for rivals—or engage users in social interactions with game world NPCs. In addition, adapting agent behaviors to foster long-term engagement stands as a key to the problem of many online games face in creating vibrant and stable communities of players.

A central challenge for lifelong agents will be adapting to games with continually evolving content in the form of patches, expansions, downloadable content, and other incremental updates to the game the agent inhabits. Research in *lifelong machine learning* investigates how agents can transfer knowledge between particular tasks, continually learn and refine knowledge, uncover representations for complex information, and incorporate guidance or feedback from humans [19]. Addressing these challenges for game agents can provide enormous benefits to developers of live and ongoing games.

*2) How can a Game AI system support deep gameplay?* A key to persistent games is player retention. Many contemporary games have high ceilings for player skills to build up to—through complex underlying systems or ever-evolving multiplayer competition. However, complex games often lose players early in the game, especially as players increasingly have more diverse backgrounds and skills. Gameplay support agents act as mentors to players to help them overcome challenges that might otherwise cause them to quit playing a game. Gameplay support AI observes players, learns their gameplay strengths and weaknesses, and intervenes to provide players with appropriate hints, training materials, or content adjustments as needed. For example, if a player shows an inability to counter a particular strategy in an RTS the AI would identify the missing player skills and could provide instruction about the appropriate response, training video demonstrations, or set up game scenarios to practice the requisite skill. At the extreme a gameplay support agent could coach players to climb their way to the top of multiplayer competition.

Intelligent Tutoring Systems [20] present one vision for gameplay support. Applying interactive tutors to support long-term player engagement will need advances in representing

player skills, modeling players based on their game activities, devising interventions to improve those skills, and adjusting those interventions in response to tutoring success or failure. Ultimately, gameplay support systems should aspire to automatically generate tutorials—for introducing new content through coaching players along difficult missions—based on game features and gameplay data. Extracting examples appropriate to demonstrate skills from gameplay data, recognizing player skills, and choosing appropriate timing and presentation style for tutorial information are all key challenges to apply gameplay data toward automated game tutoring.

Another technique for gameplay support might lie in *Dynamic Difficulty Adjustment* (DDA). Dynamic difficulty adjustment systems make real-time adjustments to game parameters, item placement, enemy behavior, and other content to suit player abilities. Techniques for DDA have involved classical cybernetic systems [21], production systems [22], optimization of generator output from neuro-evolutionary [12] or machine learning [23] systems, or logic programming on possible player behavior [8]. Adapting these techniques for gameplay support requires capturing the long-term effects of interventions on player engagement [24], richer models of player skills related to various gameplay domains, and techniques to intelligently reuse high quality human-authored content where possible.

*3) How can a Game AI system generate motivational game content?* Content generation for long-term engagement models player values, preferences, and motivations to generate content that continues player engagement over long periods of time. The goal is to improve player retention or reacquire players who have lost interest in a game. Whereas gameplay support addresses potential “pain points” of gameplay to prevent player early dropout and improve player acquisition, long-term motivational content generation focuses on how to best retain players once they have committed to a game and encourage players who have lapsed from a game to return.

Compared to previous work on content generation, motivational content generation should incentivize players to take advantage of aspects of a game they already enjoy or to explore new elements of a game they might not have encountered. For example, generation of personalized achievements can encourage players to try alternative ways to complete a mission or level. Likewise, a system might generate mini-games within the context of a larger, persistent games based on the behaviors that a player already favors. Other forms of motivational content may exist. Regardless, a system must use longitudinal player data to determine what content to create, when to create it, and what value the content will provide to different players.

Drama managers are disembodied virtual agents that monitor virtual worlds and intervene to drive a narrative forward based on models of player experience quality [4]. Drama managers implemented in *Left 4 Dead* and *Darkspore* have demonstrated the ability to increase game replay value by varying game content and modulating player intensity levels. Drama managers that procedurally generate narratives have been demonstrated in the context of short-term experiences. Extending these systems to motivate persistent game players through narratives requires further work to personalize narratives to players through data [10], create a potentially infinite variety of narratives or quests [25], and chain narratives to

create a persistent, coherent sense of progression in an ever-growing game story.

### B. Ecosystem of Games

Digital distribution and online game portals have led to increasingly diverse games and a growing long-tail of smaller games appealing to niche interests. Ecosystems of games push entertainment goals to players' experiences across a number of potentially unrelated games, rather than within a single game. Connecting characters and gameplay from a single player across multiple games opens new opportunities in cross-game agents, cross-game content generation, and automated online game designs experiments. Previous player modeling and content generation research can play new roles when players interact with many distinct games, requiring advances in representing, collecting, and reasoning on game design knowledge.

#### 4) How can AI agents interact with players across games?

Starting from the premise that lifelong virtual agents learn how to interact with individual players, we speculate that it may be advantageous for characters to interact with their players *across* many games in a company's ecosystem. To some degree, the solution involves designing for cross-game characters that manifest as recurring characters or playmates who join players across many games. However, as a lifelong agent adapts to an individual player, the question becomes one of how an agent with a constantly adapting personality, memory, and history of interactions with a player can manifest its individualized nature in the context of new games.

Cross-game agents can draw from research on competitive cross-game AI and work on socially present agents. The general game playing competition has spawned research on representing and reasoning on generic game state and rule systems to enable AI agents to play games of generic specification [26]. Cross-game agents must also consider how the personality and memory of the agent translates into new game contexts. To that end, research into *socially present agents* has explored how to control in-game behavior in reference to out-of-game context. Techniques explored include referring to historical interactions with players, simulating social roles common in a game, and explicitly signaling social and affective states not immediately relevant to in-game behavior [27]. Linking cross-game data on player-agent interactions to how social actions affect players will be crucial.

Cross-game agents may also serve as “game universe guides,” acting as a curator or tour guide to ensure players get the most out of a space of possible games. Important challenges involve understanding and eliciting player feedback on games, guiding players to new games as their interests shift, and sequencing the order of games played to optimize player experience. Cross-game data will be the linchpin to recommending different games and modeling how playing games in different orders (and ways) affects player experience.

#### 5) How can a Game AI system generate cross-game content?

To date procedural content generation systems have been developed on a game-by-game basis. Cross-game content generation and adaptation explores how data acquired about a player in one game can improve content generation in another game in the ecosystem. Cross-game data enables

mining game designs for *general design knowledge*—a model mapping game mechanics to player behavior. Taken to its extreme, such design knowledge can lead to generating new games that span genres, moving beyond the current genre-focused efforts. Understanding cross-game player behavior can also allow recommending content from other games and ultimately lead to pre-adapting game experiences to players based on their behavior in other games.

#### 6) How can a Game AI system add to the game ecosystem?

If AI as Producer focuses on an ecosystem of computer games, we might ask whether intelligent systems can automatically create new games that add to the ecosystem in meaningful ways. Computer game generation is a nascent area of Game AI research [28], [29], [30], [31]. Work to date has focused on a single genre at a time. Cross-game play puts a premium on new research to represent more generic game structures in a way that spans multiple genres. Many of the existing formalisms may be able to support such extensions, but how to best bridge genres remains an open question. More ambitious work will ultimately aspire to AI Producers that generate sets of games that complement one another, creating game ecosystems using a wealth of accumulated design knowledge.

Despite substantial efforts to reason on design knowledge, relatively little work has explored ways to acquire or refine general design knowledge. Current game industry efforts employ A/B testing for online game design—exemplified by Zynga’s practices. Analogous experimental methods have been used to understand how game designs impact player engagement and learning [32] or negative behavior [33]. Leveraging the unique potential of online distribution methods for *automated* rapid iteration and experimentation can open the way to addressing the challenges of extracting cross-game design knowledge. However, a number of research questions must first be addressed. First, an automated system must choose which designs to test, balancing benefits of testing against the high costs (in terms of player time, money, or negative reactions) of automated crowdsourced testing. Second, an automated system must be able to interpret the results of testing, including attributing credit/blame to aspects of design choices and appropriately changing designs in response. Third, an automated system must ultimately devise new design goals to explore when feedback indicates a given goal is unfeasible or unvaluable. Additionally, if user-generated content is available, a system might mine design principles from the content as a means of bootstrapping learning PCG systems.

### C. Community in Games

Game communities demand greater attention to how players impact one another’s experiences within a game, beyond the dynamics of cooperation and competition limited to a single session or round of play. Player communities extend in-game activities to a broader out-of-game social content. Careful *matchmaking* can group players for entertainment and engagement while *group-oriented content generation* can provide experiences tailored to sets of players. Further, communities themselves yield a wealth of user-generated content, posing opportunities to enhance interactions with PCG systems for better user- and system-generated content. However, with community comes a dark side: fraud, security violations, cheating, and abusive behavior. Game AI has an unique opportunity

to enhance negative behavior detection and automate responses beyond merely banning players.

*7) How can Game AI systems improve matchmaking and group content?* Matchmaking has traditionally focused on pairing players for competition to ensure even win rates. Online and social games, however, require grouping players for cooperative or synergistic goals. Beyond balancing win rates, players can be grouped to have complementary abilities or for purposes such as mentoring. Developing techniques to model player value as a social partner and using that information to create groups stand as key research challenges. Future developments will likely require adopting more sophisticated techniques from the social matching literature [34].

Group-oriented content generation extends PCG to the setting of engaging multiple players—with potentially conflicting interests—at once. Algorithms that balance the diverse needs of players in a given group, live game constraints (e.g. in terms of available players), and meet design goals are relevant research vectors. Both matchmaking and group-oriented content generation will require advances in modeling how players relate to one another socially and how these social and personal attributes interact with game content.

*8) How can a Game AI system reduce negative behavior?* Negative player behaviors in online games include fraud, security violations, cheating, and abusive behaviors. Fraudulent behavior commonly involves counterfeit game items sold for real-world money. Security violations compromise games through stealing players' accounts or financial data. Cheating spans hacking game systems to change their functionality to exploiting game bugs for personal gain or harm to others. Abusive behaviors include insulting other players or intentionally attempting to ruin their game experiences. Across these issues are a broad set of Game AI research challenges associated with responding appropriately to these behaviors. Note that many of these behaviors implicitly require cross-game agents—many forms of negative behavior manifest at the level of human players who act both within specific games and on game forums or other out-of-game socialization venues.

There is a wealth of research on fraud detection, security violations, and related challenges [35]. Little work, however, has addressed how Game AI agents should respond to these activities once detected. Game companies have only recently begun to systematically investigate ways to reduce negative behavior beyond banning players (e.g. [33]). We hypothesize that negative behavior can be reduced or mitigated by automatically adapting game content, rules, and mechanics to incentivize players toward pro-social behaviors. Game AI agents may minimize a player's negative impact by redirecting their actions away from others players. Detection may be improved by intentionally eliciting fraudulent or cheating behavior through an AI double-agent. Beyond negative reinforcement or punishment, Game AI agents can reward positive behavior or channel players toward more constructive pursuits.

*9) How can a Game AI system induce user-generated content?* User-generated content has become a major force for players' continuing interest in aging games. *Minecraft*, *Spore*, *Second Life*, and a host of other games have demonstrated the power of end-users to continually extend and enhance a live game. Despite the growing ubiquity of user-generated content

little research has developed methods to elicit needed content, interact with end-users to improve content, or mine design knowledge from this content.

Open questions for future user-generated content systems relate to how to best incorporate user content and feedback to improve the content created. Existing research has examined providing preference information (e.g. [36], [37]) or directly authoring the space of content (e.g. [29], [38], [39]). Enhancing user-generated content will require enabling users to teach PCG systems in new ways. *Interactive machine learning* has been developing techniques that optimize human abilities to train learning algorithms [40]. Integrating interactive machine learning approaches with user content creation interfaces can enable a new wave of learning PCG systems that improve through interaction with a player community. Key research problems include devising means to gather new modes of feedback, incorporating that feedback into PCG systems, and aggregating feedback from a diverse community of players.

#### D. Coupling the Real and Virtual Worlds

Games are increasingly coupled to the real world through input and output modalities that put a greater premium on a player's context. New input devices provide novel sensor information including GPS location (mobile devices), room layouts (Microsoft Kinect), motion data (Nintendo Wiimote, Playstation Move), sound (Kinect, webcams), and brain activity (NeuroSky, Emotiv). Output modalities have similarly become richer, including 3D displays (3D TV, Nintendo 3DS), second-screen experiences (Nintendo WiiU), virtual reality (Oculus Rift), augmented reality (mobile devices of all sorts), and potentially projection technologies. These technologies introduce nuances of player physical context including location, bodily motion data, and various physiological indicators. Beyond this information, games must also account for other aspects of player context including social circumstances or economic situation. Together this additional contextual information and output opportunities open research avenues related to incorporating real-world context into games, using out-of-game social network data in games and using games to proactively sense real-world information.

*10) How can a Game AI system utilize real-world context within a game?* Real-world data opens many avenues for game experiences that overlay on real-world settings or leverage real-world semantic information for new forms of gameplay. Human-computer interaction research has a rich literature on addressing the challenges and nuances of context, but has seen little adoption in the context of Game AI [41]. Understanding how AI agents can (or should) use context is an open problem for future Game AI agents.

Real-world settings enable new game types including augmented reality games and alternate reality games. *Augmented reality games* visually project virtual game content onto the real-world situation. *Alternate reality games* overlay fictional contexts on real-world settings without necessarily requiring a visual overlay—Google's *Ingress* is a prominent example. Both kinds of games, however, are currently circumscribed by challenges in authoring new content and fitting it to new, unanticipated real world contexts—a promising avenue for future PCG research. Preliminary work on merging the virtual

and real has explored dynamically adapting alternate reality game quests to new physical locations [42], [43]. The next stages of this work may use add other real world context in the game, such as social or economic context.

An alternative perspective on coupling real and virtual worlds involves using the semantic information present in real-world data. *Data Games* engage players in understanding semantic information in open data sets (e.g. US census data) through automatically mapping data into game content [44]. Future work should explore how in-game agents can leverage out-of-game data for richer interaction with players and tighter coupling of game worlds and real-world contexts.

**11) How can a Game AI system leverage out-of-game social networks?** Modeling in-game social interactions, social networks, and player communities can enable Game AI agents to interact with players as part of a social world that (partially) overlaps the in-game world. Current research on online game social networks has explored detecting social information including group identities [45], shared housing networks [46], and real-money trade [47]. Outside of games a wealth of prior work on modeling techniques for social and economic networks exists [48]. Here we ask: how can Game AI systems use social networks from outside a game to improve in-game experiences?

Overall, relating in-game interactions to out-of-game social interaction remains underexplored. Open research problems include uncovering how network structures can be used to draw users into a company’s game ecosystem or how to encourage users to move among games. Out-of-game social networks such as *Twitter* and *Facebook* additionally contain a wealth of user-generated material that reveals sentiments, preferences, attitudes, and non-game product usage. We speculate this data could be used to influence motivational content generation and integrate ads into game content in non-trivial ways.

**12) Can an intelligent system use games to “proactively sense” the real world?** In the age of big data, companies routinely collect data about users to improving products or otherwise monetizing user behavior. Yet, many aspects of player motivations or real-world context remain hidden from these efforts. *Proactive sensing* is the use of game content to elicit data about the real-world that might not otherwise be collected by passive observation of users. For example, consider attempting to learn whether a new coffee shop has good coffee. Game content—possibly a quest in an alternate reality game—can be generated that requires players within proximity of the coffee shop to visit and rate the shop, thus generating new data that might not otherwise have been generated. Additional applications of the proactive sensing paradigm can target other hidden real-world information such as player preferences, skills, and attitudes or semantic information about real-world objects and locations.

Proactive sensing research will require modeling the quality of information known about the world player abilities to provide that information, and challenges in generating game-relevant content and agent behaviors to elicit this information. Human computation research has modeled human abilities to perform tasks that provide computers with information about the real world [49]. Extending these approaches to games will involve incorporating player motivation to perform or

complete game tasks. Key research problems will include balancing between game design goals and information needs and generating game tasks and agent behaviors appropriate to a wide variety of information needs.

## V. CONCLUSIONS

The digital game industry is experiencing a shift to persistent games, business models emphasizing game ecosystems, more support for game communities, and new considerations for incorporating real world context into game worlds. We see these advances as positive signs of growth and maturity in the game industry. We also see these advances pushing the bounds on the scalability of game development practices.

Artificial intelligence, computational intelligence, and machine learning have always excelled at addressing problems of scalability by automating tasks and dynamically adapting system behavior. Game AI has always been an integral part of computer game development. AI Actors have enhanced player experiences by supporting players’ suspension of disbelief and dynamically managing dramatic contexts. AI Designers have supported and augmented the development of individual games through procedural content generation. We envision AI Producers taking on a new role of augmenting and scaling the game production pipeline, supporting the entire span of live operations in games, enhancing cross-game interoperations, nurturing strong player communities, and coupling real and virtual contexts. This vision is bolstered by the massive amounts of data being generated and collected by the game development industry.

The twelve new Game AI research questions here require game developers to see Game AI as part of live game production. AI for game production does not supplant previous challenges in Game AI—it extends the scope of the field of Game AI as a whole. Addressing these problems with credible AI solutions will result in immediate relevance to game developers and may present a new vector for industry game development and academic Game AI research collaboration.

## REFERENCES

- [1] J. Laird and M. VanLent, “Human-level AI’s killer application: Interactive computer games,” *AI magazine*, vol. 22, no. 2, p. 15, 2001.
- [2] M. Mateas, “An Oz-centric review of interactive drama and believable agents,” in *Artificial Intelligence Today*, ser. Lecture Notes in Computer Science, M. J. Wooldridge and M. Veloso, Eds. Springer Berlin Heidelberg, 1999, vol. 1600, pp. 297–328.
- [3] D. L. Roberts and C. L. Isbell, “A survey and qualitative analysis of recent advances in drama management,” *International Transactions on Systems Science and Applications, Special Issue on Agent Based Systems for Human Learning*, vol. 4, no. 2, pp. 61–75, 2008.
- [4] M. O. Riedl and V. Bulitko, “Interactive narrative: An intelligent systems approach,” *AI Magazine*, 2013.
- [5] M. Hendrikx, S. Meijer, J. Van Der Velden, and A. Iosup, “Procedural content generation for games: A survey,” *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 9, no. 1, p. 1, 2013.
- [6] J. Togelius, G. Yannakakis, K. Stanley, and C. Browne, “Search-based procedural content generation: A taxonomy and survey,” *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 3, no. 3, pp. 172–186, 2011.
- [7] G. N. Yannakakis, “Game AI revisited,” in *Computing Frontiers*, 2012, pp. 285–292.

- [8] A. M. Smith, E. Andersen, M. Mateas, and Z. Popović, “A case study of expressively constrainable level design automation tools for a puzzle game,” in *7th International Conference on the Foundations of Digital Games*, 2012, pp. 156–163.
- [9] E. Andersen, S. Gulwani, and Z. Popovic, “A trace-based framework for analyzing and synthesizing educational progressions,” in *ACM SIGCHI Conference on Human Factors in Computing Systems*, 2013.
- [10] H. Yu and M. O. Riedl, “A sequential recommendation approach for interactive personalized story generation,” in *11th International Conference on Autonomous Agents and Multiagent Systems*, 2012, pp. 71–78.
- [11] D. Thue, V. Bulitko, M. Spetch, and E. Wasylshen, “Interactive storytelling: A player modelling approach,” in *3rd AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2007.
- [12] N. Shaker, G. N. Yannakakis, and J. Togelius, “Towards automatic personalized content generation for platform games,” in *6th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2010.
- [13] M. Seif El-Nasr, A. Drachen, and A. Canossa, Eds., *Game Analytics*. Springer London, 2013.
- [14] A. Smith, C. Lewis, K. Hullett, G. Smith, and A. Sullivan, “An inclusive view of player modeling,” in *6th International Conference on the Foundations of Digital Games*, 2011.
- [15] B. Harrison and D. L. Roberts, “Using sequential observations to model and predict player behavior,” in *6th International Conference on the Foundations of Digital Games*, 2011.
- [16] B. G. Weber, M. Mateas, and A. H. Jhala, “Using data mining to model player experience,” in *FDG Workshop on Evaluating Player Experience*, 2011.
- [17] G. Yannakakis and J. Togelius, “Experience-driven procedural content generation,” *IEEE Transactions on Affective Computing*, vol. 2, no. 3, pp. 147–161, 2011.
- [18] T. Bickmore and D. Schulman, “A virtual laboratory for studying long-term relationships between humans and virtual agents,” in *8th International Conference on Autonomous Agents and Multiagent Systems*, 2009, pp. 297–304.
- [19] D. L. Silver, Q. Yang, and L. Li, “Lifelong machine learning systems: Beyond learning algorithms,” in *AAAI Spring Symposium Series*, 2013.
- [20] K. VanLehn, “The behavior of tutoring systems,” *International Journal of Artificial Intelligence in Education*, vol. 16, no. 3, pp. 227–265, 2006.
- [21] R. Hunicke and V. Chapman, “AI for dynamic difficulty adjustment in games,” in *AAAI Workshop on Challenges in Game Artificial Intelligence*, 2004.
- [22] B. Magerko, B. Stensrud, and L. Holt, “Bringing the schoolhouse inside the box - a tool for engaging, individualized training,” in *25th Army Science Conference*, 2006.
- [23] H. Yu and T. Trawick, “Personalized procedural content generation to minimize frustration and boredom based on ranking algorithm,” in *7th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2011.
- [24] A. Zook and M. O. Riedl, “A temporal data-driven player model for dynamic difficulty adjustment,” in *8th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2012.
- [25] B. Li, S. Lee-Urban, D. S. Appling, and M. O. Riedl, “Crowdsourcing narrative intelligence,” *Advances in Cognitive Systems*, vol. 2, pp. 25–42, 2012.
- [26] M. Genesereth, N. Love, and B. Pell, “General game playing: Overview of the AAAI competition,” *AI magazine*, vol. 26, no. 2, p. 62, 2005.
- [27] A. Pereira, R. Prada, and A. Paiva, “Socially present board game opponents,” in *Advances in Computer Entertainment*. Springer, 2012, pp. 101–116.
- [28] J. Togelius and J. Schmidhuber, “An experiment in automatic game design,” in *IEEE Symposium on Computational Intelligence and Games*, 2008, pp. 111–118.
- [29] A. M. Smith and M. Mateas, “Variations Forever: Flexibly generating rulesets from a sculptable design space of mini-games,” in *IEEE Conference on Computational Intelligence and Games*, 2010.
- [30] M. Cook, S. Colton, and J. Gow, “Initial results from co-operative co-evolution for automated platformer design,” in *EvoGames 2012*, 2012.
- [31] K. Hartsook, A. Zook, S. Das, and M. Riedl, “Toward supporting storytellers with procedurally generated game worlds,” in *IEEE Conference on Computational Intelligence in Games*, 2011.
- [32] D. Lomas, K. Patel, J. L. Forlizzi, and K. R. Koedinger, “Optimizing challenge in an educational game using large-scale design experiments,” in *ACM SIGCHI Conference on Human Factors in Computing Systems*, 2013, pp. 89–98.
- [33] J. Lin, “The science behind shaping player behavior in online games,” 2013, game Developers Conference. [Online]. Available: <http://www.gdcvault.com/play/1017940/The-Science-Behind-Shaping-Player>
- [34] L. Terveen and D. W. McDonald, “Social matching: A framework and research agenda,” *ACM Transactions on Computer-Human Interaction*, vol. 12, no. 3, pp. 401–434, 2005.
- [35] C. Phua, V. C. S. Lee, K. Smith-Miles, and R. W. Gayler, “A comprehensive survey of data mining-based fraud detection research,” *Computing Research Repository*, vol. abs/1009.6119, 2010.
- [36] E. Hastings, R. K. Guha, and K. Stanley, “Automatic content generation in the galactic arms race video game,” *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 1, pp. 245–263, 2009.
- [37] S. Risi, J. Lehman, D. B. D’Ambrosio, R. Hall, and K. O. Stanley, “Combining search-based procedural content generation and social gaming in the petalz video game,” in *8th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2012.
- [38] G. Smith, J. Whitehead, and M. Mateas, “Tanagra: Reactive planning and constraint solving for mixed-initiative level design,” *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 3, no. 3, pp. 201–215, 2011.
- [39] J. Dormans, “Machinations: Elemental feedback structures for game design,” in *GAMEON-NA*, 2009, pp. 33–40.
- [40] S. Amershi, J. Fogarty, A. Kapoor, and D. Tan, “Effective end-user interaction with machine learning,” in *25th AAAI Conference on Artificial Intelligence*, 2011, pp. 7–11.
- [41] J.-y. Hong, E.-h. Suh, and S.-J. Kim, “Context-aware systems: A literature review and classification,” *Expert Systems with Applications*, vol. 36, no. 4, pp. 8509–8522, 2009.
- [42] A. Macvean, S. Hajarnis, B. Headrick, A. Ferguson, C. Barve, D. Karnik, and M. O. Riedl, “WeQuest: Scalable alternate reality games through end-user content authoring,” in *8th International Conference on Advances in Computer Entertainment Technology*, 2011, p. 22.
- [43] A. Gustafsson, J. Bichard, L. Brunnberg, O. Juhlin, and M. Combetto, “Believable environments: generating interactive storytelling in vast location-based pervasive games,” in *ACM SIGCHI International Conference on Advances in Computer Entertainment Technology*, 2006, p. 24.
- [44] M. G. Friberger, J. Togelius, A. B. Cardona, M. Ermacora, A. Mousten, M. Jensen, V. Tanase, and U. Brøndsted, “Data games,” in *4th Workshop on Procedural Content Generation*, 2013.
- [45] C. Grappiolo, J. Togelius, and G. N. Yannakakis, “Interaction-based group identity detection via reinforcement learning and artificial evolution,” in *GECCO Evolutionary Computation and Multi-agent Systems and Simulation Workshop*, 2013.
- [46] M. A. Ahmad, B. Keegan, D. Williams, J. Srivastava, and N. Contractor, “Trust amongst rogues? a hypergraph approach for comparing clandestine trust networks in MMOGs,” in *Fifth International AAAI Conference on Weblogs and Social Media*, 2011, pp. 17–21.
- [47] A. Fujita, H. Itsuki, and H. Matsubara, “Detecting real money traders in MMORPG by using trading network,” in *7th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2011, pp. 26–31.
- [48] M. O. Jackson, *Social and economic networks*. Princeton University Press, 2010.
- [49] E. Law and L. Ahn, “Human computation,” *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 5, no. 3, pp. 1–121, 2011.



# Approaches to Measuring the Difficulty of Games in Dynamic Difficulty Adjustment Systems

Dagmara Dziedzic & Wojciech Włodarczyk

To cite this article: Dagmara Dziedzic & Wojciech Włodarczyk (2018): Approaches to Measuring the Difficulty of Games in Dynamic Difficulty Adjustment Systems, International Journal of Human-Computer Interaction, DOI: [10.1080/10447318.2018.1461764](https://doi.org/10.1080/10447318.2018.1461764)

To link to this article: <https://doi.org/10.1080/10447318.2018.1461764>



Published online: 20 Apr 2018.



Submit your article to this journal 



Article views: 12



View related articles 



View Crossmark data 



# Approaches to Measuring the Difficulty of Games in Dynamic Difficulty Adjustment Systems

Dagmara Dziedzic<sup>a</sup> and Wojciech Włodarczyk<sup>b</sup>

<sup>a</sup>Department of Logic and Cognitive Science, Institute of Psychology, Adam Mickiewicz University, Poznań, Poland; <sup>b</sup>Faculty of Mathematics and Computer Science, Information Systems Laboratory, Adam Mickiewicz University, Poznań, Poland

## ABSTRACT

In this article, three approaches are proposed for measuring difficulty that can be useful in developing Dynamic Difficulty Adjustment (DDA) systems in different game genres. Our analysis of the existing DDA systems shows that there are three ways to measure the difficulty of the game: using the formal model of gameplay, using the features of the game, and direct examination of the player. These approaches are described in this article and supplemented by appropriate examples of DDA implementations. In addition, the article describes the distinction between task complexity and task difficulty in DDA systems. Separating task complexity (especially the structural one) is suggested, which is an objective characteristic of the task, and task difficulty, which is related to the interaction between the task and the task performer.

## 1. Introduction

For years, the difficulty was an integrated element of the game design process (Schweizer, 2006). First attempts to manipulate the difficulty of the game emerged with the launch of first gaming platforms and the coin-op arcade games in the 1970s. It was the time when games began to give the player an opportunity to choose the level of difficulty from one of the predefined options. As mentioned by Schweizer (2006), the first popular video game in which a player had a choice between two options (beginner's race and advanced player's race) was SpeedRace – an arcade game produced in 1974 by the Taito Corporation (1974). Soon after, game developers went one step further and created a first video game in which the difficulty was adjusted to player's doings. This game was the Heart of Africa (1985), in which the key locations were moved if the player had a problem with finding them (Brathwaite & Schreiber, 2008).

Today, the ability to adjust the game's difficulty to the needs and experience of a player is associated with the ability to choose one of several available options (e.g., easy, medium, hard). The choice between these options or their counterparts began to be commonly used in games in the 1990s (e.g., in Sid Meier's Civilization, 1991; Doom, 1993; Jazz Jackrabbit 2, 1998) and is still used today (we can find it in the latest productions, e.g., in Call of Duty, 2003; The Witcher, 2007; Dishonored, 2012; Sid Meier's Civilization VI, 2016). Regardless of which difficulty the player chooses, the difficulty of the game will usually be further increased with the progress – the further the player proceeds, the more difficult the game becomes.

In the examples described earlier, the player decides which difficulty of the game he/she wants to choose, but his/her choice is limited just to few options that were previously prepared by game designers. In addition, the difficulty of the game is always incremented at the fixed point in a game (e.g., when a player reaches a certain level). This approach is an example of the game-centered game design, in which human experts (i.e., designers) independently determine the parameters of the game (see Hristova, n.d.). In this type of game design, difficulty curve is static and is not influenced by the behavior of the player during the game. This approach treats every player equally and ignores differences in their abilities and preferences. This way, in some periods of the gameplay, difficulty may not be matched to the skills of the players (Yun, Shastri, Pavlidis, & Deng, 2009).

Recently, game designers tend to choose an alternative approach – the player-centered game design (or the user-centered game design), in which the games are designed, so that they can meet the expectations of players. One of the essential elements of player-centered game design is to provide an appropriate level of challenge, a properly matched learning curve and an enhancement of the gameplay experience for each player (Charles & Black, 2004). In this approach, developers design the so-called Dynamic Difficulty Adjustment (DDA) systems to ensure proper difficulty for each individual player.<sup>1</sup> Their purpose is to modify some elements of the gameplay, so that the general difficulty will be changed in the response to a player's performance, or his/her affective states. Measurement of difficulty in the DDA is a

**CONTACT** Dagmara Dziedzic  [dagmara.dziedzic@amu.edu.pl](mailto:dagmara.dziedzic@amu.edu.pl)

<sup>1</sup>What is more adjusting the difficulty should be done without interrupting the game, so the player does not feel cheated by the game. Effective difficulty adjustment should take place without degrading player's experience and influencing his/her sense of control and accomplishment. For more about player's enjoyment and satisfaction, see Hunicke (2005) and Andrade, Ramalho, Gomes, and Corruble (2006).

case of measuring the difficulty in which the measurement directly affects the adjustment. The goal is not to measure the difficulty by itself, but rather focus on the issue of how it allows the system to adjust the difficulty of the game to the player.

This approach is very promising, because game design, where the difficulty is adjusted to the player's skills can bring many benefits. The problem of difficulty in the game is important for both practical and theoretical reasons. From a practical point of view, in this model, game developers are able to increase their profits and reach more customers. However, the design of DDA systems applies not only to commercial productions – this approach can be also easily implemented in games with a purpose, serious games, educational games, and rehabilitation games. Games with challenges adjusting to the skills of players can reach a larger audience, be more interesting for the players and educate them more effectively. For example, Hamari et al. (2016) report that educational games in which challenges increases with the growing skills and knowledge of students can efficiently engage them in learning activities.

From a theoretical point of view, the researchers of the difficulty in games (intentionally or not) are trying to answer questions that go beyond the realm of games, such as What is the difficulty? Can the difficulty be defined? How a person feels and perceives the difficulty? How the difficulty can affect the motivation and satisfaction?

Currently, there is no consistent definition of the difficulty that would be used by all designers of DDA systems. There is also no inclusive approach to the problem of difficulty and the possibility of adjusting it. In addition, in existing literature, concepts of difficulty and complexity are not distinguished and often used interchangeably.

This article is an attempt to present existing ways of measuring the difficulty of games for purpose of DDA systems. Within the article, we distinguish three types of difficulty measurement techniques. The DDA designer creating a new system can choose the technique which suits the characteristics of game he/she is working on. Moreover, the article attempts to address the issue of distinguishing the complexity and difficulty of games.

The article describes in [Section 2](#) – purpose of DDA systems, in [Section 3](#) – difficulty measurement techniques in DDA systems, and in [Section 4](#) – the difference between the concepts of game's complexity and game's difficulty.

## **2. Purpose of DDA systems**

As mentioned in [Section 1](#), the implementation of DDA systems is the realization of player-centered game design. Its main goal is to adjust the gameplay to the skills of the individual player. To achieve this, the designers of DDA often rely on the flow theory, which suggests how well-designed automatic difficulty adjustment should work (regardless of the genre of games). Flow concept was

introduced by Csikszentmihalyi (1975) who describes it as a “holistic sensation that people feel when they act with total involvement” (p. 36). Flow is something felt by people who are totally involved in a task they are performing, and this task is done for the pure enjoyment.

The state of flow can be successfully achieved by the players. However, to achieve it, appropriate conditions have to be met. According to Nacke (2012) in the case of games, the player must first perform a challenging activity that requires him/her to improve his/her skills. Second, the activity performed by him/her should have clear and easy to achieve goals (levels, missions, high scores, etc.) with an immediate feedback on his/her progression. Third, the final result of the activity the player is performing should be uncertain, but at the same time, he/she has to have a direct impact on its outcome.

One of the most important conditions for experiencing the flow is to provide the challenges that the player perceives as adequate to his/her abilities. Missura (2015) points out that an integral feature of any challenge (and the learning process required to master it) is its difficulty. The difficulty is a subjective factor, which changes (decreases) with time and effort that the player spends on learning the skill required to accomplish a certain task (Missura, 2015).

The flow theory suggests that the difficulty of the game should offer challenges that difficulty corresponds to the current skills of the player. If the player feels that the challenges in the game become too difficult for him/her too quickly – he/she experiences frustration (Falstein, 2005) or even an anxiety (Csikszentmihalyi, 1975). If the difficulty of the challenges don't change – and the skill of the player rises – the player experiences a boredom (Csikszentmihalyi, 1975). The path between these two extremes is referred to as the flow zone.

One of the solutions, which may be used to ensure that the player stays in the flow zone, regardless of his/her experience and skills is using DDA systems. Depending on the genre of the game, difficulty is handled in a different way, using different parameters of the game.

## **3. Difficulty measurement techniques in DDA systems**

While creating a DDA system, a designer has to define a method for measuring the difficulty and conditions describing how it will be manipulated. After an analysis of existing DDA systems, one may realize that depending on various types of games, the difficulty can be manipulated using different elements of the game, also the intervention of a DDA system (which increases or reduces the overall difficulty of the game) may occur in different points during the game.<sup>2</sup> Therefore, these properties of DDA systems are usually imposed by the structure of the game itself (for more general discussion, see Dziedzic, 2016). Even though there is no single, generally accepted definition of the difficulty, designers are able to

<sup>2</sup>The analysis was carried out with four search engines for scientific articles: Google Scholar (<https://scholar.google.com>), EbscoHost (<https://search.ebscohost.com>), ACM Digital Library (<http://dl.acm.org/>), ScienceDirect (<http://www.sciencedirect.com/>), using a combination of keywords such as *dynamic, difficulty, adjustment, DDA, games*.

create a DDA system using simplified, informal metrics based on the characteristics of the game they are working on.

Examples of the DDA systems created in this way are listed in [Tables 1–3](#). These are examples of measuring and/or adjusting the difficulty of games that allowed us to identify the three general approaches used to measure the difficulty. Each example includes the definition of difficulty proposed by the game authors, difficulty measurement (which describes in detail how the aforementioned definition was implemented), and the way in which difficulty was adjusted (if it was included in the research). Examples of DDA systems outlined in [Tables 1–3](#) not only allowed us to distinguish three approaches to measure the difficulty but also showed that some DDA systems are based primarily on game complexity (see [Section 4](#)).

The difficulty measurement techniques described later define three approaches that can be used while developing a DDA system. The first two are directly related to gameplay, and the third is player-related. The first technique involves using a formal model, which can be defined as a detailed description of rules of the game (one of the consequences of having a formal model of the game is the ability to construct its game tree<sup>3</sup>). The second technique is used when the formal model of the game is unknown (or highly complicated). In such case, DDA systems use the features of the game to create an approximate measurement for the real difficulty. We distinguish two types of features: parameters that describe the selected elements of the game (e.g., number of opponent's pawns, number of enemies, number of lost lives, number of collected coins) and player's performance indicators that do not have direct representation in the game (e.g., the number of enemies killed per minute, average rate of gain of coins). In what follows, the terms "parameters" and "indicators" will be understood as described earlier. Apart from using gameplay-related techniques, creators of DDA systems can independently analyze the player himself/herself. In the third technique of measuring difficulty, DDA systems are evaluating the feeling of the difficulty perceived by the player. The measurement of difficulty involves querying the player about his/her perceived difficulty, or measuring emotions he/she felt during the game. This measurement can take place independently of the gameplay, so it can be mixed with the first two difficult measurement methods.

To sum up, one can define three distinct approaches to measure the difficulty: (1) the use of formal model of gameplay, (2) the use of selected features of the game, and (3) the direct examination of the player.

### **3.1. Using the formal model of gameplay**

The use of this technique occurs in board games such as chess or connect four (see [Table 1](#)). These are games in which the player chooses his/her next move from a finite

set of moves possible in his/her current state of the game and the gameplay is divided into units of time (turns).

Within this group, DDA systems are based on the analysis of the structure of the game tree. The number of possible moves at every moment is relatively small. Performance of the player (whether the player is going to win or lose) is evaluated based on the analysis of his/her movements – and more specifically possible consequences of his/her moves retrieved from the game tree. DDA systems working as described previously are most often based on search algorithms such as MiniMax or Alpha-beta Pruning (see [Missura & G'Artner, 2008](#)). Algorithms of this type operate on the basis of the evaluation function of each player's movement. Typically, this function analyzes the consequences of player's move as a change in the game's parameters before and after player's action (e.g., change in the number of enemy pawns). For example, in chess, capturing opponent's pawn can be considered as a good move which leads to winning, and loss of pawn as a bad move that increases the chances that the player will lose. Using this knowledge, the system changes the current difficulty by selecting opponent's moves which difficulty matches those done by the player (the same measure is used to evaluate both player's and opponent's moves). One of the most important advantages of this approach is that even if game trees of different games can vary a lot, they can all be analyzed in the same manner. Regardless of whether a DDA system is designed for games like Tic-Tac-Toe or chess, it can use evaluation function based on the structure of the game tree such as the shortest path to success, or the average path length to failure. Intervention of the DDA system, created for this type of games, can be done in units of time naturally imposed by the game – after each round. Exemplary systems of the described type are presented in [Table 1](#). In chess by Guid and Bratko ([2013](#)) and "Mastermind" by Gerasimczuk, Der Maas, and Rajmakers ([2013](#)), authors have proposed only a measure of difficulty, without including a DDA system in their research. However, both solutions show a good example of difficult measurement techniques for games with a formal model. Due to the fact that adjusting difficulty in various games with a game tree can be fairly similar, creating a measure of difficulty in this type of games is more important than creating an adjustment algorithm that could be transferred from one game to another.

Main assumption that one has to accept if he/she chooses to use this approach is that the structure of the game can reflect the performance of the player and influence the way he/she perceives the difficulty (features such as the length of the shortest path to success determines the difficulty of the game).

### **3.2. The use of selected features of the game**

The second technique occurs in games, where the formal model is not known or it is too complicated to use it – due to a large

<sup>3</sup>Game tree refers to the concept of a tree known from the graph theory. In this tree, every possible state of the game is represented as a node in a graph. And if a player can move from a given state to another in exactly one move, then these states are connected with an edge. The final game states are represented as leaves of the tree. Each play of the game is represented as a path leading from the root to one of the leaves (for more details, see [Gobe, De Voogt, & Retschitzki \(2004\)](#)).

**Table 1.** The use of a formal model of the game in exemplary implementations.

| The use of formal model of the game                  |  |  |  |
|--|--|--|--|
| Game title and source                                | Definition of difficulty used in the research  | Difficulty measurement   | Adjustment   |
| Connect Four by Missura and Gärtner (2008)           | "The problem of an <b>automatic difficulty scaling</b> can be viewed then in a context of an interaction between a player and one (or more) in-game agent... It is natural to assume that at any given time the agent has a set of actions (strategies) available to it. The question of how to adjust the game difficulty automatically can then be formulated as what action (strategy) should an in-game agent choose as next." (Missura & Gärtner, 2008, p. 1)   | Authors proposed a search-based algorithm called AdaptiveMiniMax (AMM). <ul style="list-style-type: none"> <li>(1) Using a full game tree, MiniMax algorithm is creating a set of all possible moves together with their evaluation.</li> <li>(2) Score of each move depends on length of its path to winning or losing (it gets more points if player can win faster).</li> </ul>   | (1) Algorithm evaluates moves done by the player using this metric and creates a ranking of them.<br>(2) AMM adjusts the difficulty of the game by selecting moves that matches an average score from the ranking. |
| Chess by Guid and Bratko (2013)                      | "We focus our investigation on problems in which the <b>difficulty</b> arises from the combinatorial <b>complexity of problems</b> . We propose a <b>measure of difficulty</b> that is based on modeling the problem solving effort as search among alternatives and the relations among alternative solutions." (Guid & Bratko, 2013, p. 860)   | Authors proposed an search-based algorithm for measuring difficulty of chess tactical problem. Proposed metric was created using following principles: <ul style="list-style-type: none"> <li>(1) If its solution requires many steps (solution lies deep in the game tree), the problem is considered difficult.</li> <li>(2) The problem is more difficult if the possible solutions are located far from each other in the game tree.</li> </ul>  | N/A  |
| Mastermind (simplified) by Gerasimczuk et al. (2013) | "We associate the <b>difficulty</b> of Deductive Mastermind <b>game-items with the size</b> of the corresponding logical trees obtained by the tableau method." (Gerasimczuk et al., 2013, p. 297)<br><br>"Normatively speaking, the full tree generated by the tableau method for the set of formulas corresponding to a DMM-item, represents an adequate reasoning scheme. Therefore the <b>size of the tree</b> (e.g. the number of nodes) can be thought of as an <b>abstract complexity measure</b> ." (Gerasimczuk et al., 2013, p. 307) | The authors developed a model for calculating the difficulty of the game, which is based on the logical reasoning, using analytic tableaux. <ul style="list-style-type: none"> <li>(1) The algorithm contains a list of logic formulas that are used for reasoning. Formulas describe basic conclusions that can be made from the game's feedback.</li> <li>(2) The algorithm creates a tree (similar to game tree) with the states, which can be reached using the logical reasoning.</li> <li>(3) The difficulty of the game is computed as a size of the tree.</li> </ul> | N/A  |

number of moves the player can make it is not possible to create a full game tree. Games of this type often include arcade elements, random events or time-related tasks. Examples of games using this technique belong to such genres as first-person shooters, tower defense, platformers, etc. (see Table 2).

Because it is not possible to determine all the possible moves of the player, in this method, the model of the difficulty is reduced only to a closed group of features. In this model, DDA can use both parameters that describe the selected elements of the game (e.g., number of enemies) and indicators of player's performance (e.g., number of enemies killed per minute). Because these features are very dependent on the game's genre and the gameplay elements, each game has its own unique set of features, manually selected by the DDA designer.

Also, the intervention time of the system depends strictly on the characteristics of the game for which DDA is designed. If it is a game with distinct stages of gameplay (e.g., waves of enemies in tower defenses), the intervention occurs after each stage. Otherwise, if the game does not contain such stages, the

intervention must occur in some fixed intervals (e.g., every 30 s).

DDA systems using this technique are based on various algorithms. Recently, one of the most popular approaches is to use machine learning algorithms (such as clustering algorithms – see Lora, Sánchez-Ruiz, González-Calero, & Gómez-Martín, 2016) to determine the relation between players performance and the difficulty of the game.<sup>4</sup> Exemplary systems, which use this technique of measuring the difficulty, are described in Table 2.

Main assumption that one has to accept if he/she wants to use this approach is that selected features are adequate and sufficient to measure and manipulate the difficulty felt by the player.

### 3.3. Direct examination of the player

In this method, designers of DDA systems analyze the difficulty perceived by the player instead of elements of the gameplay. Therefore, this approach can be applied to any game,

<sup>4</sup>Machine learning algorithms are designed so that for a set of input features algorithm assigns a result value (numerical for regression algorithms or class for classification algorithms). Machine learning works well in this approach because the difficulty model is also based on a set of features. A DDA creator using machine learning must prepare a training data with specific values of the features and a specific difficulty level assigned to them. Then the learning algorithm will try to find the relationship between the features and the level of difficulty. One of the most popular uses of machine learning in games is creating an artificial intelligence (AI) that plays with the player (e.g., AlphaGo created by DeepMind, <https://deepmind.com/>). While AI algorithms mentioned earlier is designed to simply win with the player, the purpose of the DDA is to adapt to his/her performance. For more details about machine learning, see Smola and Vishwanathan (2008).

**Table 2.** The use of selected parameters of the game in exemplary implementations.

| The use of selected parameters of the game                      |   |  |   |
|---|---|--|---|
| Game title and source   | Definition of difficulty used in the research   | Difficulty measurement   | Adjustment  |
| Hamlet (first-person shooter) by Hunicke and Chapman (2004)     | "Using techniques drawn from Inventory Theory and Operations Research, Hamlet <b>analyzes and adjust the supply and demand of game inventory</b> in order to control overall <b>game difficulty</b> ." (Hunicke & Chapman, 2004, p. 96)<br>"We propose a probabilistic technique that dynamically evaluates the <b>difficulty</b> of given obstacles based on user performance, as the game is running." (Hunicke & Chapman, 2004, p. 97)   | Authors proposed a probabilistic model for measuring difficulty in the game using player's inventory. The algorithm uses the following rules to determine the current performance of a player.<br>(1) Algorithm observe player's health and distribution of damages taken by player over time to predict potential health shortfalls.<br>(2) Algorithm analyzes changes in player's inventory. How much he gains (e.g., ammunition found in creates) and losses during the game (e.g., used ammunition). | Authors created a number of adjustment policies with adjustment actions and cost estimations.<br>(1) System support two types of actions: reactive (adjusting elements that are "on stage", e.g., entities that have notices the player) and proactive (adjusting elements that are "off state" e.g., spawning health).<br>(2) System is determining the cost of a given action to make sure that player do not notice system's intervention. |
| Tower Defense by Sutoyo, Winata, Oliviani, and Supriyadi (2015) | "... we determine the <b>difficulties</b> based on players' <b>lives, enemies' health, and passive skills</b> ... that are chosen by the player. With three of these factors, players will have varies experience of playing tower defence because different combination will give different results to the system and difficulties of the games will be different for each gameplay." (Sutoyo et al., 2015, p. 435)  | Author selected three parameters that are monitored to determine difficulty of the game through player's performance.<br>(1) Algorithm analyzes player's lives, enemies' health and passive skills (skill points).<br>(2) Passive skills are three groups: offensive, defensive and support.   | (1) DDA modifies the set of parameters (such as the number of gold or spawned opponents) and their multipliers (e.g., gold multiplier).<br>(2) The DDA system adjusts the difficulty by modifying the aforementioned parameters after each round (e.g., decreases the number of opponents).   |
| Tetris by Lora et al. (2016)                                    | "When a player starts a new game we look at his first movements to find the most similar cluster. Then the system provides dynamic help to the player choosing "good" Tetris pieces from time to time. In our experiments, using DDA users obtain higher scores and report improvements in terms of their game experience." (Lora et al., 2016, p. 335)<br>"The <b>difficulty</b> in Tetris depends mainly on two parameters: The <b>type of pieces</b> and the <b>falling speed</b> ." (Lora et al., 2016, p. 337) | The authors created a statistical clustering model that is used to estimates player's skills.<br>(1) Performance of the player is calculated based on his/her last 10 tactical decisions (placing the block on the map).<br>(2) Each decision is described by a set of parameters (e.g., height of the blocks, scored points, maximum number of points to win).<br>(3) Using collected data players were divided into three groups: newbie, average, expert.   | (1) Algorithm is assigning player to a one of aforementioned groups (algorithm is updated after each move).<br>(2) Depending on player's group, algorithm helps him/her frequently (for newbie) or infrequently (for experts). Algorithm helps the player by giving him block needed in current state of the game.  |

regardless of whether it has a formal model or not (see Table 3).

The difficulty measure of the game is based on the **player's feelings**. The specific feelings of a player are correlated with the various difficulty settings of the game. When a measuring tool recognizes that the game is too hard for the player (or too easy), the DDA system adjusts the difficulty.

Just like in the previous case, this technique does not impose the moment of intervening – as it depends directly on the characteristics of the game.

Examples of methods used for measuring the difficulty are questionnaires measuring affective states and physiological tests. Example systems, which use this technique of measuring the difficulty, are described in Table 3. All the examples cited in this table apply only to research carried out with the use of games, and not commercial games. Even though it is hard to apply this technique in commercial projects, it can provide the most reliable information about the perceived difficulty directly from the player. Furthermore, it is possible to use this technique to assign difficulty to appropriate gameplay parameters (e.g., the number of enemies) based on difficulty perceived by the player. At the end of the study, collected data can be used to create a statistical adjustment model that does

not require polling (or testing) the player, and may already be used in commercial games.

Main assumption that one has to accept if he/she chooses to use this approach is that the player is able to determine the difficulty of the task and compare it with the difficulty of other, and that selected affective or physiological states are directly related to perceiving the difficulty of the game.

### 3.4. Conclusions

Due to the fact that it is difficult to find a game where it is possible to implement all three techniques of measuring difficulty in DDA systems and to test which gives better results, it is impossible to decide which of the above could be consider as a best one. Choosing the right technique highly depends on the structure of the gameplay and the purpose of the DDA system. Table 4 lists the practical suggestions that can help to select the right difficulty measurement technique for a new DDA system. It provides a summary of approaches described in this section, their advantages, disadvantages, and the suggested types of games for each of the techniques.

**Table 3.** Direct examination of the player in exemplary implementations.

| Direct examination of the player                |  |  |   |
|---|--|--|---|
| Game title and source                           | Definition of difficulty used in the research  | Difficulty measurement   | Adjustment  |
| Third-person shooting game by Yun et al. (2009) | "The measurements are based on the assumption that the players' performance during the game-playing session alters blood flow in the supraorbital region, which is an indirect measurement of increased mental activities." (Yun et al., 2009, p. 2195)<br>"... we propose a system that adaptively and automatically adjusts the <b>game difficulty level</b> based on measurements of the <b>players' facial physiology</b> ." (Yun et al., 2009, p. 2196) | Authors proposed a method for measuring difficulty by analyzing the player's changes in the skin temperature in the supraorbital region using the StressCam.<br>(1) The change in temperature in this region is associated with the blood flow, which is an indirect measure of the mental activities.<br>(2) When the game is hard and the player suffers stress or frustration – the temperature in the supraorbital region should grow. | (1) Authors created three game modes: easy, moderate, difficulty. Each mode is associated with different strength of enemies robots.<br>(2) Game difficulty was altered during the game using the metric described earlier.   |
| Pong by Liu, Agrawal, Sarkar, and Chen (2009)   | "In this DDA mechanism, a <b>player's physiological signals</b> [e.g. skin conductance, skin temperature, heart sound] were analyzed to infer his or her probable <b>anxiety level</b> , which was chosen as the target affective state, and the <b>game difficulty level</b> was automatically adjusted in real time as a function of the player's affective state." (C. Liu et al., 2009, p. 506)  | The authors created an affect based DDA, which recognize the affective state (anxiety) using psychophysiological analysis.<br>(1) To develop affective model, the authors built mappings from physiological features to the intensity of anxiety.<br>(2) Low anxiety means that the player perceives the game as easy and high anxiety level means that the game is to hard.   | (1) Real-time player's anxiety level recognition was used to adjust the game's difficulty.<br>(2) To manipulate the difficulty level of the game, authors selected a number of parameters (e.g., ball speed and size, paddle speed and size, sluggish or over-responsive keyboard, random keyboard) and adjusted them during the game.                                  |
| Runner Game by Medeiros and Medeiros (2014)     | "... we propose a way to automatically rank all combinations of features according to how fun that level is, and by doing so, choose the combination of features with the best flow for our runner game. After each 30 second playthrough, the <b>player is asked</b> to rate <b>how fun</b> and <b>how difficult</b> was that level on a scale from 1 to 5." (Medeiros & Medeiros, 2014, p. 797)  | Authors proposed a system in which the difficulty measurement is done by the player himself after each round of the game.<br>(1) Player plays the game for 30 seconds or until avatar's death.<br>(2) After the game, he/she completes a survey, in which (on a scale from 1 to 5) he/she describes how fun and difficult the game was.  | (1) The authors defined several features of the game (e.g., number of spikes per wave, distance between spikes) that are used to control the difficulty of the game.<br>(2) The system selects a set of features for each game. Every time the player rated a particular configuration to be good/bad – the algorithm changes the probability of current configuration. |

#### 4. Complexity versus difficulty

Analyzing at how the difficulty is defined in games with a full game model (Table 1), one may notice that it is modeled using only game's parameters and actions related to them (e.g., capturing opponent's pawn as a decrease in the number of opponent's pawns on the map). In particular, the difficulty of these games is measured mainly on the basis of their game tree – core of DDA systems do not take into account the indicators of player's performance (how quickly he/she puts the pawns, how logically he/she reasons), nor his/her emotions and physiological states.

If so, the difficulty measurement for games with a full game model corresponds to struturalist viewpoint of task complexity proposed by Liu and Li (2012). These authors provided a comprehensive review and conceptualization of task complexity and tried to determine the differences between task complexity and task difficulty.

Further in this section we rely primarily on their work because it is a holistic overview based on the analysis of several dozen research studies in which the task complexity appeared. Their approach has not been previously applied to games, but in our opinion the complexity of the game can be considered as a special case of much more general task complexity – and more specifically task complexity in structuralist viewpoint.

According to Liu and Li (2012), in structuralist approach, task complexity is defined as the structure of the tasks and it may be computed as a function of the number of elements that form the task and the relations between these elements (e.g., number of elements, number of targets, number of solutions, number of tracks, number of alternatives).

In the case of the complexity and the difficulty, there are no definitions which would clearly explain these concepts, as a consequence, they are often confused. In accordance with Liu and Li (2012), we claim that in the case of games:

To avoid the misuse and misunderstanding of these related terms, we claim that task complexity can be distinguished from task difficulty in that task complexity involves the objective characteristics of a task, whereas task difficulty involves the interaction among task, task performer, and context characteristics. Subjective task complexity is a perceived complexity by the task performer. In general, task difficulty refers to the extent to which task performers feel difficulty in performing a task. (p. 559)

The term difficulty includes not only the complexity of the task but also the performer who performs it and the context of the task. If so, then the complexity of the game is only a subset of the difficulty, and to create a more complete game difficulty model for games with a full model, DDA designers should consider incorporating the aspect of the task performer (player) into their systems.

DDA systems created for games with a full model operate on the basis of objective gameplay characteristics and they depend only on actions of a player (and not his/her skills or emotions). Liu and Li (2012) suggested that objective task complexity is independent of task performer and it is closely related to task characteristics. By looking at the definitions in Table 1 (compared to Tables 2 and 3) only in the first one, the authors define the difficulty based solely on the task complexity. The definitions imposed by the authors are determined only by the structure of the problem that player has to solve (as in the structuralist task complexity by Liu & Li, 2012). In this approach, the role of the performer is limited or does not appear at all. In addition, taking the performer into account can be a challenge, because in logic games it is difficult to capture concepts such as logical reasoning (which affects the next move of the player) in a measurable way. In the game with a full game model, DDA system analyzes only the final effect of the player's actions (e.g., the fact of capturing the pawn on the board in chess is more important than the number of pawns captured per minute).

In the games where the formal model of the game is unknown (or highly complicated), complexity is also present but it cannot be described in a simple way (e.g., due to the limited computing power of current computers). Hence, it is necessary to simplify the game to a closed set of measurable features. In games of this type, DDA system analyzes the sequence of player's actions spread over time (e.g., the time needed for the player to aim at the enemy). The task performer's aspect is measured indirectly with the player's performance indicators, which directly reflect the player's skill (such as hand-eye

coordination). The task difficulty seems to be more appropriate for describing a game in which the task complexity is not determined by the formal model and the aspect of the task performer is taken into account.

In the case where the task performer's aspect is measured directly with the player's examination, the presence of the performer's task is evident – regardless of whether the game contains a full model or not, this technique is based on task difficulty. A summary of what constitutes the base of the difficulty measure for each difficulty measurement technique is given in Table 4.

The conclusion that complexity is only a subset of difficulty can also be reached by considering situations in which the same complexity of tasks may have different difficulties for different performers. For example, for most children learning the multiplication tables, multiplying "six times eight" is far more difficult than the multiplication of other numbers. Multiplying "six times ten" and "six times eight" have the same structural complexity, but in the latter case, the children respond poorly in 62.5% of cases (Magazine Monitor, 2014).

Another case, in which the same structural complexity of the task gives a different feeling of the difficulty is chess. In studies conducted by Hristova, Guid, and Bratko (2014), chess experts evaluated the difficulty of chess tactical problems. For this purpose, authors selected 12 positions – chess tactical problems (selected randomly from Chess Tempo<sup>5</sup>), which were divided into three classes of difficulty (easy, medium, hard). This division was made on the basis of the Chess Tempo rating – ranking, which is based on the success and failures of the players (if the player solves the problem correctly, its rating goes down, and if incorrectly –

**Table 4.** Comparison of difficulty measure techniques in DDA systems.

| Technique name                              | The use of formal model of gameplay  | The use of selected features of the game  | The direct examination of the player  |
|---|--|---|---|
| Difficulty measure                          | Based on the game tree   | Based on the game parameters and player's performance indicators  | Based on the difficulty perceived by the player   |
| Base of difficulty measure                  | Task complexity (task performer's aspect is limited or does not appear)                          | Task difficulty (task performer's aspect is measured indirectly with player's performance indicators)   | Task difficulty (task performer's aspect is measured directly with the player's examination)  |
| Difficulty adjustment method                | Changes the difficulty by selecting moves of the opponent using a measure based on the game tree | Changes the parameters of the game affecting the difficulty   | Changes the parameters of the game affecting the difficulty   |
| Advantages                                  | DDA systems created for different games share similar algorithms (e.g., MiniMax)                 | The possibility to create DDA systems for complex games by narrowing them to a set of measurable features   | The player directly reports his or her feelings of difficulty rather than indirectly, through the values of the selected metrics  |
| Disadvantages                               | Requires a full game model   | Large range of parameters and indicators that are the basis of DDA systems for different games (there can be multiple, different DDA systems created for the same game) | Creators of DDA system need an access to the player (large interference in the player's environment) and to have the tools to measure the player's, e.g., declared difficulty, affective states or physiological states |
| Typical moment of DDA system's intervention | After each game's turn   | Depends on the structure of the game (the moment is chosen by the designer or after each turn in the game)  | Depends on the moment when the player is questioned and the game structure (the moment is chosen by the designer or after each turn in the game)  |
| Suggested game types                        | Board games, logic games   | Arcade games, shooter games, racing games, platformers  | Arcade games, shooter games, racing games, platformers  |

<sup>5</sup>The Chess Tempo is an online chess platform available at [www.chesstempo.com](http://www.chesstempo.com).

its rating goes up). Chess experts were asked to solve these 12 tactical problems, then rate them from 1 to 12 (they did not know that the problems had previously been divided into three classes of difficulty using the Chess Tempo rating). The study included a retrospective reports, which would help to understand the approach that experts used while solving a certain position. The results showed a large disproportion between Chess Tempo rating, and the estimations of experts. Sometimes, the difficult chess problem was considered easy by the experts and easy problem as hard. The same arrangement of pieces on a chessboard (structural complexity) produced a different result in the assessment of the difficulty made by the experts.

The same complexity of gameplay can result in different difficulty for different players. Therefore, DDA systems that do not take into account aspects of the player will not be able to fully adapt to the specific player. The problem of dependency between complexity and difficulty in games is still open and there is currently no research showing how these two concepts affect each other.

## 5. Conclusions

For years, designers have created games that give gamers a choice of gameplay difficulty. In the most basic approach, players can choose between easy, medium, and hard mode of game. However, this may be insufficient because the players' skills and experience are so different that it is difficult to grasp them just with the use of a finite number of categories. In addition, the player learns during the game, so his/her skills and expectations of the difficulty may change over time. The solution to this problem is to design DDA systems that are based on the flow theory. According to it, the challenge must be adjusted to the player's skill, so he/she can constantly enjoy the game. DDA systems are designed to measure difficulty (based on their difficulty conception), and adapt it.

However, existing DDA systems are missing a coherent approach to measure the difficulty. After an analysis of existing implementations of DDA systems, we have proposed a difficulty measurement techniques, which can be applied in different game genres. The approaches presented in this article distinguish three types of difficulty measures: using the formal model of gameplay, using the features of the game, and direct examination of the player. The first two are separable and relate to gameplay itself, while the third one is related only to the player and can be used independently of the first two (together with them or completely separate).

The designer creates a DDA system for full model games, and it should definitely use it to adjust the difficulty. But taking into account conclusions from Section 4, he/she may consider extending the difficulty model and including the performer's performance (e.g., by incorporating the techniques of measuring difficulty from direct examination of the player). If the designer creates a DDA for a game without a full model, then he/she should use a closed group of game parameters along with indicators of player's performance to adjust the difficulty. The selection of features depends on the game genre and gameplay elements. As basic features, it is important to choose the ones that define the key elements of the gameplay: e.g., in the

shooter games – the number of killed enemies, in tower defence – the number of lives, and in racing games – speed of the car and player's ranking position.

In addition, this article describes the distinction between task complexity and task difficulty in DDA systems. We suggest separating task complexity (especially the structural one), which is an objective characteristic of the task, and task difficulty, which is related to the interaction between the task and the task performer. In DDA systems, which use the full model of gameplay (e.g., chess), the adjustment mechanism is based on the complexity of the gameplay (usually, the structure of the game tree, which is an objective characteristic of gameplay). Since this issue (to our best knowledge) has not been noticed before, there is currently no research that would show the influence of task complexity on the perceived (subjective) difficulty or enjoyment felt by player.

## References

- Andrade, G., Ramalho, G., Gomes, A. S., & Corruble, V. (2006). Dynamic game balancing: an evaluation of user satisfaction. In J. E. Laird & J. Schaeffer (Eds.), *AIIDE* (pp. 3–8). Menlo Park, CA: The AAAI Press.
- Arkane Studios. (2012). *Dishonored* [PC game]. Bethesda Softworks.
- Brathwaite, B., & Schreiber, I. (2008). *Challenges for game designers* (1st ed.). Rockland, MA, USA: Charles River Media, Inc.
- CD Projekt RED. (2007). *The Witcher* [PC game]. Atari, CD Projekt.
- Charles, D., & Black, M. (2004). *Dynamic player modelling: A framework for player-centered digital games*. Proceedings of the international conference on computer games: Artificial intelligence, design and education (p. 29–35). Reading: Microsoft Campus.
- Csikszentmihalyi, M. (1975). *Beyond boredom and anxiety*. San Francisco, CA: Jossey-Bass Publishers.
- Dziedzic, D. (2016). Dynamic difficulty adjustment systems for various game genres. *Homo Ludens*, 9(1), 35–51.
- Epic MegaGames. (1998). *Jazz Jackrabbit 2* [PC game]. Gathering of Developers, Logicware, Project Two Interactive.
- Falstein, N. (2005). Understanding fun - the theory of natural funativity. In S. Rabin (Ed.), *Introduction to game development*. Hingham, MA: Charles River Media.
- Firaxis Games. (2016). *Sid Meier's Civilization VI* [PC game]. 2K Games.
- Gerasimczuk, N., der Maas, H. L. J. V., & Rajmakers, M. E. J. (2013). An analytic tableaux model for deductive mastermind empirically tested with a massively used online learning system. *Journal of Logic, Language and Information*, 22(3), 297–314. doi:10.1007/s10849-013-9177-5
- Gobe, F., de Voogt, A., & Retschitzki, J. (2004). *Moves in mind: The psychology of board games*. Hove, East Sussex: Psychology Press.
- Guid, M., & Bratko, I. (2013). *Search-based estimation of problem difficulty for humans*. Artificial intelligence in education - 16th international conference, AIED (pp. 860–863). Memphis, TN. doi: 10.1007/978-3-642-39112-5\_131
- Hamari, J., Shernoff, D. J., Rowe, E., Coller, B., Asbell-Clarke, J., & Edwards, T. (2016). Challenging games help students learn: An empirical study on engagement, flow and immersion in game-based learning. *Computers in Human Behaviour*, 54, 170–179. doi:10.1016/j.chb.2015.07.045
- Hristova, D. (n.d.). *Dynamic difficulty adjustment (DDA) in first person shooter (FPS) games*. Retrieved April 05, 2017 from [https://www.academia.edu/4921794/Dynamic\\_difficulty\\_adjustment\\_DDA\\_in\\_First\\_person\\_shooter\\_FPS\\_games](https://www.academia.edu/4921794/Dynamic_difficulty_adjustment_DDA_in_First_person_shooter_FPS_games)
- Hristova, D., Guid, M., & Bratko, I. (2014). Assessing the difficulty of chess tactical problems. *International Journal on Advances in Intelligent Systems*, 7(3), 728–738.
- Hunicke, R. (2005). *The case for dynamic difficulty adjustment in games*. Proceedings of the 2005 ACM SIGCHI international conference on advances in computer entertainment technology (pp. 429–433). New York, NY: ACM.

- Hunicke, R., & Chapman, V. (2004). AI for dynamic difficulty adjustment in games. *Challenges in Game Artificial Intelligence AAAI Workshop* (pp. 91–96). San Jose, CA.
- id Software. (1993). *Doom [PC game]*. GT Interactive.
- Infinity Ward. (2003). *Call of Duty [PC game]*. Activision, Aspyr.
- Liu, C., Agrawal, P., Sarkar, N., & Chen, S. (2009). Dynamic difficulty adjustment in computer games through real-time anxiety-based affective feedback. *International Journal of Human-Computer Interaction*, 25(6), 506–529. doi:10.1080/10447310902963944
- Liu, P., & Li, Z. (2012). Task complexity: A review and conceptualization framework. *International Journal of Industrial Ergonomics*, 42(6), 553–568. doi:10.1016/j.ergon.2012.09.001
- Lora, D., Sánchez-Ruiz, A. A., González-Calero, P. A., & Gómez-Martín, M. A. (2016). *Dynamic difficulty adjustment in Tetris*. Proceedings of the twenty-ninth international florida artificial intelligence research society conference, FLAIRS (pp. 335–339). Key Largo, FL.
- Magazine Monitor. (2014). *Who, what, why: Why does the sum 7 × 8 catch people out?* Retrieved April 05, 2017, from <http://www.bbc.com/news/blogs-magazine-monitor-28143553>
- Medeiros, R. J. V. D., & Medeiros, T. F. V. D. (2014). *Procedural level balancing in runner games*. Proceedings of the 2014 Brazilian symposium on computer games and digital entertainment (pp. 109–114). Washington, DC: IEEE Computer Society.
- Missura, O. (2015). *Dynamic difficulty adjustment* (Doctoral dissertation). University of Bonn. Retrieved April 05, 2017, from <http://hss.ulb.uni-bonn.de/2015/4144/4144.pdf>
- Missura, O., & Gärtner, T. (2008). *Online adaptive agent for connect four*. Proceedings of the fourth international conference on games research and development cybergames (pp. 1–8).
- MPS Labs. (1991). *Sid Meier's Civilization [PC game]*. MicroProse.
- Nacke, L. E. (2012). *Flow in games: Proposing a flow experience model*. Proceedings of the workshop on conceptualising, operationalising and measuring the player experience in videogames at fun and games (p. 104–108). Toulouse, France: ACM.
- Ozark Softscape. (1985). *Heart of Africa [Commodore 64 game]*. Electronic Arts.
- Schweizer, B. (2006). Difficulty. In H. Lowood & R. Guins (Eds.), *Debugging game history: A critical lexicon* (pp. 109–117). Cambridge, MA: MIT Press.
- Smola, A. J., & Vishwanathan, S. (2008). *Introduction to machine learning*. Cambridge, MA: Cambridge University Press. Retrieved from <http://alex.smola.org/drafts/thebook.pdf,/bib/smola/smola2008ml/thebook.pdf>
- Sutoyo, R., Winata, D., Oliviani, K., & Supriyadi, D. M. (2015). Dynamic difficulty adjustment in tower defence. *Procedia Computer Science*, 59 (1), 435–444. doi:10.1016/j.procs.2015.07.563
- The Taito Corporation. (1974). *Speed Race [Arcade game]*. Author.
- Yun, C., Shastri, D., Pavlidis, I., & Deng, Z. (2009). *O'game, can you feel my frustration?: Improving user's gaming experience via StressCam*. Proceedings of the SIGCHI conference on human factors in computing systems (pp. 2195–2204). New York, NY: ACM. doi: 10.1145/1518701.1519036

## About the Authors

**Dagmara Dziedzic**'s research is associated with the possibility to make games used in the study, which were a useful tool for researchers and an attractive entertainment for the player. She is also working on the use of user experience techniques to improve gameplay enjoyment.

**Wojciech Włodarczyk** focuses on designing and creating crowdsourcing systems used for acquiring linguistic resources. His scientific interests are also connected with creating artificial intelligence agents in computer games, as well as applying machine learning mechanisms in order to improve them.

# DYNAMIC DIFFICULTY ADJUSTMENT IN VIDEO GAMES

## Title

Empirical evaluation of player experience using a machine-learning approach to dynamic difficulty adjustment in video games.

## Author names and affiliations

Nigel Robb<sup>a</sup>, Bo Zhang<sup>b</sup>

<sup>a</sup> University of Tokyo.

<sup>b</sup> East China Normal University.

## Corresponding author

Nigel Robb

Center for Global Communication Strategies, University of Tokyo, Japan

[nigelrobb@g.ecc.u-tokyo.ac.jp](mailto:nigelrobb@g.ecc.u-tokyo.ac.jp)

*Empirical evaluation of player experience using a machine-learning approach to dynamic difficulty adjustment in video games.*

*Abstract*

Dynamic difficulty adjustment (DDA) in video games involves altering the level of challenge provided based on real-time feedback from the player. Some approaches to DDA use measurements of player performance, such as success rate or score. Such performance-based DDA systems aim to provide a bespoke level of challenge to each player, so that the game is neither too hard nor too easy. Previous research on performance-based DDA shows that it is linked to better player performance, but finds mixed results in terms of player experience (e.g., enjoyment). Also, while the concept of flow is regarded as an important aspect of video game experience, little research has considered the effects of performance-based DDA on flow. We conducted an experiment on the effects of performance-based DDA on player performance, enjoyment, and experience of flow in a video game. DDA was achieved using a generalised algorithm. 221 participants played either the DDA version of the game, a control version (difficulty remained constant), or an incremental version (difficulty increased regardless of performance). Results show that the DDA group performed significantly better. However, there were no significant differences in terms of enjoyment or experience of flow.

*Keywords*

video games; dynamic difficulty adjustment; game balancing; flow; performance; adaptive software

## 1 Introduction

Most video games involve some challenge for the player. Some games are generally easy, some are generally hard, and almost all games feature some change in the difficulty level over time; typically, games get harder the further the player progresses (Chang, 2013). In this paper, we refer to this traditional approach as “incremental difficulty adjustment”. Furthermore, a diverse range of people (e.g., in terms of age, gender, disability, motivations, and preferences) play games (Entertainment Software Association, 2017; Williams et al., 2008). Taken together, these points begin to illustrate the complex challenges involved for the game designer when determining the difficulty of a game, to enhance the experience for a range of players. If the game is too easy, more skilled players may be bored; but if it's too hard, less skilled players may be frustrated (Leiker et al., 2016). It is likely that this applies regardless of the genre or intended purpose of the game. Whether it is a fast-paced action game intended to entertain, a puzzle game for mathematics education, a cognitive training game for children with cognitive impairments, or even a language-learning application with game-like features, it is obviously essential that players engage optimally with the software to ensure the desired outcome. As such, the level of challenge provided by a game is an important consideration.

Within this broad issue, one interesting potential solution to some of these challenges lies in dynamic difficulty adjustment (DDA). DDA in video games refers to any technique in which

## DYNAMIC DIFFICULTY ADJUSTMENT IN VIDEO GAMES

the difficulty of the game is altered during the game (or perhaps between games) in response to some feedback about the player's experience (Xue, et al., 2017). The aim is to continuously tailor the difficulty of the game to each individual player. DDA has featured in video games since at least 1981 (Adams, 2008). The promise of DDA lies in the fact that the game designer does not need to pre-determine one specific difficulty curve (or even a range of pre-determined curves, as in games which let the player select, e.g., Easy, Medium, or Hard mode). Instead, the designer can effectively provide a range of possible difficulties which are dynamically selected for each individual player based on their experience of the game. Essentially, this provides a bespoke difficulty curve for each player.

### *1.1 Approaches to dynamic difficulty adjustment*

One important distinction when using DDA is based on the metric used to provide feedback for game adaptation. In this paper, we have so far (intentionally) characterised this feedback very broadly, as "player experience". In practice, player experience can be determined in a variety of ways. We can broadly categorise approaches to DDA in two ways, depending on whether they use players' in-game performance to provide the feedback, or use some information about the players affective state. A combination of these approaches would of course also be possible.

Performance-based DDA involves measuring the player's performance in the game, and adjusting the level of challenge provided accordingly. For example, the survival horror game Left 4 Dead generates a unique experience for individual players by tailoring – among other things – the enemies encountered by the player, based upon measurements of player performance (Booth, 2009). In other words, as the player performs better, the game gets harder, and vice versa. Previous research on performance-based approaches to DDA demonstrates the wide range of choice available in the design of such systems, both in terms of the indicator of player skill (i.e., the feedback), and the game features that are subsequently adjusted. Regarding the measurement of player skill, previous approaches have used, for example, the time taken to complete a task (Sharek & Wiebe, 2015), players' scores (Bateman et al., 2011), or, in more complex systems, multiple measurements may be combined and evaluated to determine the current state of the player (Hunicke, 2005). Regarding the game features that are adjusted, these range from simple adjustments such as changing the speed of the game (Alexander et al., 2013) or changing the number of objects a player must interact with (Nagle et al., 2015; Robb et al., 2019), to more complex systems which alter the behaviour or characteristics of computer-controlled enemy characters (Hunicke, 2005) or dynamically generate the layout of the game environment (Shaker et al., 2010).

Affective DDA refers to any approach which aims to use some feedback about the player's emotional state as the basis for the adaptivity. A growing body of research has investigated the feasibility of using psychophysiological measurement to provide an index of players' emotions during gameplay, and adapt the game experience based on these measurements. Measures used include cardiovascular (e.g., heart rate), electro-dermal activity (i.e., galvanic skin response, which is directly dependent on sweat-gland activity), electromyography (which measures muscle movements), and neuroimaging techniques. The latter category includes techniques such as electroencephalography and functional near-infrared spectroscopy; both of which use sensors attached to the head to provide real-time measurements of brain activity with a relatively high temporal resolution (Thibault et al.,

2016). These techniques have been used to adapt game difficulty, for example, by increasing the speed of the game or decreasing the size of targets. In addition, some studies have used affective feedback to alter other features of a game not related to difficulty, such as lighting and audio effects. For a comprehensive review of research in this area and references for all examples discussed in this paragraph, see Bontchev (2016). One obvious disadvantage of affective-based DDA is the requirement for additional equipment (which is often large and expensive) to obtain the psychophysiological feedback. Therefore, affective DDA is most likely not yet suitable for widespread use in video games. However, technological advances will undoubtedly address this issue. For example, preliminary work shows the potential of using machine vision techniques to infer players' emotional states based on real-time data obtained from the front-facing camera in smartphones and tablets (Bevilacqua et al., 2015; Bevilacqua et al., 2016). However, due to this current limitation of affective DDA, we will focus on performance-based DDA in the remainder of this paper. We will, however, return to the issue of affective DDA in Section 4, and show how the system used in the present study may also be used with affective feedback.

### *1.2 Effects of performance-based dynamic difficulty adjustment*

Previous research has investigated the effects of using performance-based DDA on various aspects of the game playing experience. Several studies in this area have considered the relationship between DDA and player enjoyment. Alexander et al. (2013) used an experimental game to investigate how DDA compared with incremental difficulty adjustment. They found that, while casual gamers reported enjoying a simple 2D game more when the difficulty was dynamically adjusted according to their performance, experienced gamers (who made up most of the sample) enjoyed the game more when the difficulty was adjusted incrementally. This study also showed that players enjoyed the game more when the difficulty was tailored to their gaming experience (casual vs experienced) rather than their performance. However, in a sample of 90 players, only 19 were categorised as casual players. Furthermore, this classification was determined by the players' response to the question "are you a casual or experienced gamer?"; it is therefore difficult to determine how to understand the distinction between casual and experienced gamers as the classification criteria are not explicit.

Nagle et al. (2015) also found that performance-based DDA led to lower player enjoyment than an alternative system in which players could control the level of difficulty throughout the game themselves. They created a 3D game in which players had to memorize a list of objects and locations, then find the objects and place them in the correct location. The difficulty of the game was determined by the number of objects (more is harder) and the number of times they could view the list of objects and location numbers (fewer is harder). However, while player enjoyment was lower with DDA, DDA was associated with better player performance. That is, when they allowed players to control the number of objects and number of times the list could be consulted, performance was significantly lower than when these values were automatically adjusted based on player performance.

Sharek & Wiebe (2015) also showed that DDA was associated with better player performance. Using an isometric puzzle game, they created over 100 different levels which were tested and ordered by difficulty. They had three difficulty conditions: DDA, in which more difficult levels were provided to players based primarily on measures of performance; incremental; and a choice condition, in which players were given the option to select a

## DYNAMIC DIFFICULTY ADJUSTMENT IN VIDEO GAMES

harder or easier level after each completed level. Players in the DDA condition showed significantly higher performance (indexed as reaching more difficult levels more quickly) than players in the other two conditions (see also Baldwin et al., 2014).

However, in a study by Orvis et al. (2008), no significant differences in performance or motivation were found between 4 groups, playing versions of a game with either no difficulty adjustment, incremental difficulty adjustment, or adaptive adjustment (two versions, distinguished in terms of the starting difficulty, which was either easy or hard). Other research on motivation finds similar negative results, with DDA not associated with significantly different levels of player motivation than incremental difficulty adjustment in a Spanish language education game (Sampayo-Vargas et al., 2013). However, in line with previous results showing increased performance, the authors showed that DDA led to significantly higher learning outcomes, which they attribute to a scaffolding effect wherein the reductions in difficulty (the “scaffold”) are provided when students need support, then removed when students were ready to progress.

While Sharek & Wiebe (2015) claim that DDA increases performance with no decrease in engagement, Altimira et al. (2017) found that DDA increased player engagement in a digitally augmented game of table tennis. By projecting images onto a surface, they could increase or decrease the difficulty of the game (e.g., by altering the size of the virtual table projected onto the surface). They found that adjusting the difficulty in response to the score differentials between the two players (e.g., by making one player’s half of the table smaller, thus making the game harder for the opposing player), was associated with significantly higher player engagement (self-report questionnaire) than no adjustment. Preliminary results from research by Xue et al. (2017) using DDA in a mobile game distributed via the Google Play Store and Apple App Store show that DDA increases player engagement over a longer period (4 months). This study is notable as it measures player engagement objectively, in terms of total time spent playing the game. The authors also note that using DDA had no effect on the amount of revenue generated from in-game transactions.

Altimira et al. (2017) also highlight another potential application of DDA technology, in that they showed that using DDA significantly reduced the score differences between players, thus allowing less skilled players to be more competitive against more skilled players. Other studies have successfully used DDA to reduce skill differentials between players of different abilities (e.g., Jensen & Grønbæk, 2016; Hwang et al., 2017). Gerling et al. (2014) created a rhythm game (i.e., in which players must perform steps in time with music) which could either be controlled by a dance mat (i.e., the player inputs the rhythm with their feet), buttons on a standard video game controller, or a via a wheelchair input (wheelchair movements are captured by a motion sensor camera). Using various techniques, they produced adaptive versions of the game which allowed less-skilled able-bodied players to compete with more-skilled able-bodied players, and players with and without mobility disabilities to compete together. Bateman et al. (2011) also decreased performance differentials between players by providing adaptive targeting assistance in a simple shooting game. DDA may therefore be important for enabling people with disabilities to play multiplayer games with those without disabilities (Hernandez et al., 2013; Hwang et al., 2017). DDA may also be one factor which can increase the effectiveness of rehabilitation games for people with disabilities, both in terms of making such games accessible and in terms of adapting the difficulty of the games to suit players of a wide range of abilities and

provide an optimum level of challenge, which is shown to increase the effectiveness of such games (Barrett et al., 2016; Hocine et al., 2015).

To summarise, previous research generally supports the idea that performance-based DDA can increase player performance, and that it can be used to reduce performance differentials between players of different abilities. Proposed applications of this include making games more accessible or inclusive for people with disabilities, and, related to this, increasing the effectiveness of games for rehabilitation. However, in terms of player experience – i.e., engagement, enjoyment, and motivation – previous research provides mixed results on the effects of DDA.

### 1.3 Flow

Several issues relevant to DDA are captured in the concept of flow, which was first introduced by Csikszentmihalyi (1975). During a flow state, an individual is completely focused on an activity; it is an enjoyable and fulfilling experience (an “optimal” experience), often described colloquially as being in “the Zone” (Chen, 2007). Csikszentmihalyi identifies several characteristics of the flow state, including having clear goals, concentrating on the task at hand, feeling in control, and being engaged in a challenging activity requiring skill. Flow was originally modelled by Csikszentmihalyi as an optimal balance between anxiety and boredom; the zone in which one is challenged enough to not be bored, but skilled enough to not be anxious about one’s performance (Csikszentmihalyi, 1988). Since Csikszentmihalyi’s original work, a large body of research has further investigated and characterised flow, and several instruments have been developed for measuring the subjective experience of flow (Weibel et al., 2008). Some of this work has explicitly focused on video games, which have been claimed to “possess ideal characteristics to create and maintain flow experiences in that the flow experience of video games is brought on when the skills of the player match the difficulty of the game” (Sherry, 2004). The importance of the flow experience is shown by empirical research suggesting that flow is one of the reasons why people play video games (Perttula et al., 2017). In educational games, flow has been used as a measure of game quality, and a small amount of research suggests that the experience of flow is associated with the effectiveness of game-based learning (Perttula et al., 2017).

The notion of a balance between player skill and game difficulty shows the direct relevance of flow to performance-based DDA. If the aim of these approaches to DDA is to match the difficulty of a game to each unique player’s skill level, then it seems likely that DDA could be used to achieve a balance between these two factors, and therefore encourage flow experiences. However, although theoretical discussions of flow feature in much research on DDA, we are aware of only one previous study investigating the relationship between difficulty adjustment and flow empirically. Using a modified version of *Tetris* and a within-subjects design with 30 participants, Ang & Mitchell (2017) showed that playing with incremental difficulty adjustment and with a version of DDA in which the player could control the difficulty were both associated with significantly different scores (compared to no DDA) on several constructs of the Flow State Scale (Jackson & Marsh, 1996). This included challenge-skill balance, which was greater when participants played a game with DDA. Note, however, that the DDA used in this study is not performance-based as discussed in this paper, in that players manually (and voluntarily) increased or decreased the difficulty

## DYNAMIC DIFFICULTY ADJUSTMENT IN VIDEO GAMES

during the game themselves, rather than have the difficulty automatically adjusted based on a measurement of player performance.

### 1.4 Machine learning for a generalised adaption mechanism

One of the limitations of many previous DDA systems is their specificity. Much of the research discussed in this paper makes use of context-specific algorithms; that is, the algorithm is designed and implemented for the game at hand, based on “domain-specific requirements” (Chang, 2013). In other words, many previous systems rely on a heuristic that only applies to either a specific game, or a specific genre of games. This limits the extent to which these systems can be readily applied to other games, and limits the applicability of research findings using these systems.

The machine learning approach to DDA developed by Missura & Gärtner (2011) is intended to address this issue. Their partially ordered set master (POSM) algorithm is a generalised DDA mechanism which does not rely on a domain-specific heuristic. A detailed description of the algorithm is beyond the scope of this paper, but the procedure can be summarised as follows. Given a set of possible difficulty settings, some will be too hard for the player and some will be too easy. Furthermore, because difficulty settings can be ordered (i.e., they can be sorted from least to most difficult), it follows that if a setting is judged to be too easy (or too hard), then all settings easier (or harder) than it will also be too easy (or too hard).

Based on this, the POSM algorithm operates by allowing the player to play one round of the game (this can be a level, or a period of any duration); next, based on feedback from the game (expressed simply as “too hard” or “too easy”), the algorithm updates its “belief” (a numerical value) about the suitability of that setting, and all settings easier (or harder) than that setting. The algorithm then selects the setting which currently has the highest belief value as the most appropriate for the player in the next round (see Missura & Gärtner, 2011, for a detailed description of the POSM algorithm). The advantage of this approach is that the algorithm is blind to both what the settings are (they could be game speed, number of enemies, or anything that can be quantified) and the heuristic that determines the feedback; in other words, the message “too hard”/“too easy” is sent from the game to the POSM algorithm, and deciding which message to send is the responsibility of the game, not the POSM algorithm. This shows how POSM is generalised: it selects settings based on feedback, but it does not know what those settings are or how the feedback is determined. As such, it should be possible to easily apply POSM in almost any game, without modification. However, as far as we are aware, POSM has only been implemented and evaluated (with human players, as opposed to simulations) in turn-based strategy games (Illici et al., 2012). Therefore, the feasibility of such an approach to DDA in other kinds of video game remains to be investigated.

### 1.5 The present study

Our aim in this study was to investigate how a performance-based approach to DDA based on POSM affects player performance, enjoyment, and experience of flow, using an experimental game created for this study. We conducted a controlled experiment with 3 groups, with each group playing a different version of the game. The independent variable was the way in which the difficulty of the game was adjusted, with 3 levels: (1) DDA, (2) incremental difficulty adjustment (in which the game gets progressively harder irrespective of player experience), and (3) no difficulty adjustment (control group). In the DDA version of the game, we used a generalised machine learning algorithm (based on POSM) to adjust the

## DYNAMIC DIFFICULTY ADJUSTMENT IN VIDEO GAMES

difficulty, and a measure of player performance provided the feedback upon which the adjustments were based.

### 1.5.1 *Hypotheses*

Our hypotheses were (1) DDA will produce greater player performance than either incremental or no difficulty adjustment; (2) DDA will lead to greater experience of flow than either incremental or no difficulty adjustment; and (3) DDA will lead to greater enjoyment than either incremental or no difficulty adjustment.

## 2 *Materials and methods*

### 2.1 *The game*

To test our hypotheses, we designed and implemented a simple video game called “Meteor Shower” (Figure 1). The object of the game is to avoid the meteors which continually fall from the top of the screen while catching the pink falling stars. The player controls the yellow character by moving left or right along the bottom of the screen (using the left and right directional arrows on the keyboard). The velocity of the meteors determines the difficulty of the game, with higher velocities making the game more difficult (i.e., the meteors are harder to avoid). The game consists of 20 levels, each lasting 45 seconds, with a short pause between each level. Players are awarded one point for each star they catch, and lose a point each time a meteor hits the character. The score is reset to 0 at the end of each level. There are 8 falling stars to catch in each level (each falling 5 seconds apart), and so the maximum score available on any level is 8 (i.e., the player catches all stars and avoids all meteors). While the meteors and stars appear to the player to originate from random locations, the game in fact uses a seeded random number generator to ensure that the pattern of locations at which stars and meteors appear is the same for each player. When generated, stars fall in a straight line. Meteors move in a straight line from their point of origin, to the location of the player-controlled character at the time the meteor was generated. This ensures that players always must move the character to avoid every meteor. Three versions of the game were created. The sole difference between each version is the way in which the difficulty (i.e., the velocity of the meteors) was adapted during gameplay.

## DYNAMIC DIFFICULTY ADJUSTMENT IN VIDEO GAMES

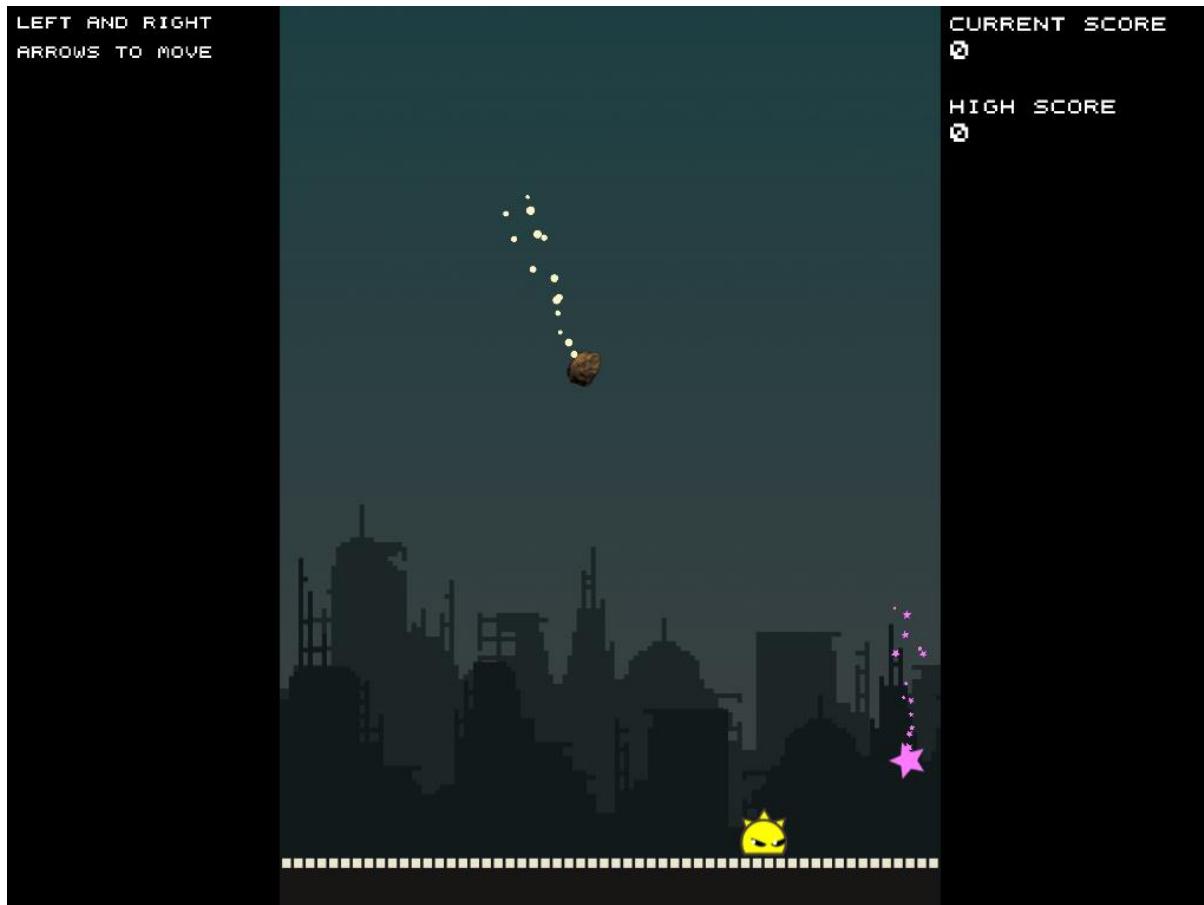


Figure 1. The experimental game, Meteor Shower. The player moves the yellow character left and right using the directional arrows on a keyboard. The meteor, which is moving towards the yellow character, should be avoided. The pink star, which is falling in a straight line, should be collected. The velocity at which the meteor falls dictates the difficulty of the game (faster is more difficult).

### 2.1.1 Control version of the game

In this version of the game, the velocity of the meteors remained constant at 800 throughout.

### 2.1.2 Incremental version of the game

In this version of the game, the velocity of the meteors increased by 50 at the start of each new level. The velocity on level 1 was 200; the velocity on level 20 was 1150.

### 2.1.3 DDA version of the game

This version of the game used a machine learning algorithm, based on the POSM algorithm, to adapt the velocity of the meteors in response to measurements of player performance. The starting value on level 1 was 900. The possible settings ranged from 200 to 1700, increasing in increments of 50. Due to an error in the programming, value 1200 was not used.

As described in Section 1.4, the POSM algorithm requires a message to be regularly sent from the game describing if the current setting is too hard or too easy for the player. In this study, this message was sent every 5 seconds. The decision was based on two measures of player success rate. The first of these measured the player's success at avoiding meteors over the previous 5 seconds of play ( $success1 = (nM - nH)/nM$ , where  $nM$  is the number of meteors in the previous 5 seconds and  $nH$  is the number of times the player was hit by a

meteor in the previous 5 seconds). The second measurement was the player's overall success rate based on their score and the potential maximum score possible at that point in the level ( $\text{success2} = \text{score}/nS$ , where  $nS$  is the number of stars that have fallen so far in the level).

Determining which value to use for the target success rate (i.e., the value above which the game was judged to be too easy), proved to be one of the most challenging design decisions in the development of the game, and we were unable to find previous research to guide this decision. Therefore, during the development process, we played versions of the game using success rates ranging from 0.75 (i.e., 75%) to 1 (i.e., 100%). We determined that the most satisfying experience was provided when we used a target success rate of 1 for both measures.

## 2.2 Questionnaire

Flow was assessed using the Flow Short Scale (Rheinberg et al., 2003), which is shown to be a reliable measure of flow experience in video games (Wiebel et al., 2008). We used the online version of the scale (<http://www.psych.uni-potsdam.de/people/rheinberg/messverfahren/fks1-e.html>) which has 14 items. Items 1 – 10 measure flow, items 11 – 13 measure anxiety, and item 14 measures perceived skill demands (challenge) (see Appendix in Engeser, 2012). In addition, demographic information (age, gender, frequency of video game play) was collected.

## 2.3 Performance data

During gameplay, each participant's score was recorded for levels 1 – 19. Due to a bug in the software (which we identified after running the experiment), the score for level 20 was not recorded. For the DDA group only, we also recorded the velocity of the meteors at 5 second intervals (i.e., each time the velocity was updated). This provided a measure of the difficulty of the game (higher velocity is more difficult), and a measure of the player's skill level (better players will reach higher velocities). Scores were recorded for all participants in the DDA group for 855 seconds of gameplay (i.e., not the full 15 minutes, due to a technical problem or bug, currently unidentified). We did not record velocity for the control group, as this remained constant throughout the game (800) or for the incremental group, as this increased on a pre-determined scale with each level.

## 2.4 Data collection method: Amazon Mechanical Turk

We conducted the experiment online using Amazon Mechanical Turk (MTurk). MTurk has been described as a “marketplace for work that requires human intelligence” (Rouse, 2015). Users of the site are classified as “requesters” (who post tasks to the site) and “workers” (who complete these tasks in return for payment). Example tasks include completing surveys to provide feedback about a website, classifying images based on their content, or providing translations of short pieces of text. Typically, MTurk is appropriate for tasks which can be completed quickly, and for which many instances of the task must be completed. MTurk is now frequently used to conduct research in psychology (Paolacci & Chandler, 2014; Mason & Suri, 2012), and it has been used in at least one previous study on the effects of DDA in video games (Sharek & Wiebe, 2015).

However, several issues have been identified which may threaten the validity of data obtained from MTurk (Cheung et al., 2017). Some of these issues are not unique to MTurk. For example, the issue of selection bias, in the sense that MTurk workers choose the tasks

they wish to complete, applies in any research wherein participants voluntarily opt to take part after viewing, e.g., a poster or advertisement on social media. However, all participants in research conducted via MTurk will (obviously) have self-selected to become MTurk workers. Related to this, Cheung et al. (2017) note that samples obtained from MTurk may not be representative of the population of interest (e.g., all MTurk participants are internet users, which not be appropriate for some studies). On this issue, we make two observations. Firstly, we note that, in some respects, samples obtained from MTurk may be more diverse than samples obtained by traditional means. Casler et al. (2013) point out that most participants in psychological research are American college students; they showed that a sample of MTurk workers was significantly, desirably more diverse in terms of ethnicity, economic status, and age, than a sample of undergraduate students. Secondly, we point out that the nature of our study – in which participants are required to play an online video game remotely – dictates that participants would be required to have internet access and be reasonably computer literate whether recruited through MTurk or not.

Perhaps the most important concern with MTurk data highlighted by Cheung et al. (2017) is the possibility of participants answering questions without paying attention to the content (either fully, or at all, i.e., selecting random answers). However, steps can be taken to mitigate this risk (see also Fleischer et al., 2015). Firstly, MTurk incorporates a rating system for workers, so that requesters can specify that only workers of a suitable quality can access their tasks. We will discuss this further in Section 2.6, where we describe how we used this system to specify that only workers of a certain quality could access our experiment. Secondly, items can be included in questionnaires to check for attentiveness (e.g., a multiple-choice item that states which option the respondent should select). Finally, it may also be possible to detect inattentive responses by analysing data gathered, although this would presumably depend on the nature of the data. The screening process we used to identify inattentive participants in the present study is described in Section 2.7.

As should be the case in all data collection processes, we recommend that consideration is given to potential limitations of data collected using MTurk, and that these limitations are addressed as fully as possible.

## 2.5 Procedure

The task was made available using MTurk's Survey Link template. This provides a link to an external website, where participants complete a task (typically a questionnaire) and receive a completion code. They then enter the completion code in MTurk to receive credit for completing the task. The default configuration of the Survey Link template allows each worker to only complete the task once. In our case, the survey link took participants to a site hosting the game. Which version of the game was loaded was determined randomly by the software. Participants pressed a button to start the game when they were ready. After 15 minutes of play, the game automatically ended, and participants were presented with a link to the webpage containing the questionnaire. When they submitted the questionnaire with all questions completed, the data were stored on a server, and a unique completion code was generated on the server and returned to participants. The completion code, a record of which version of the game they had played, and performance data automatically recorded during gameplay, were also stored on the server. Participants then entered the completion code in MTurk, and were paid \$0.99. All participants were paid, whether their data were included in the analysis or not.

## 2.6 Pilot

Initially, we ran a pilot with 10 participants. By considering participants' performance data, it appeared that some participants did not actually play the game. It would be possible to merely let the game run for 15 minutes, then select random answers to the questions. To address this, we included two attention check items in the questionnaire. These items stated that the participant should select option 7 ("very much") in the Likert-style scale. We also screened the data collected during the full experiment and removed data which appeared to show no engagement with the game (see Section 2.7). In addition, we decided to use MTurk's qualifications feature to ensure that the task was only available to workers who (1) had completed over 10000 tasks on MTurk; and (2) and an approval rate of 97% or higher.

## 2.7 Participants

Data were collected from 300 adults via MTurk, each randomly assigned to play either the control, DDA, or incremental versions of the game. Entries in which answers to either of the attention check questions were incorrect were removed. We also removed entries in which a score of zero was recorded for every level, as this suggested that the participants did not actually play the game, but merely let it run for the required time. Finally, we considered the velocity data recorded for participants in the DDA group, and removed any entries in which the pattern of velocities across the levels suggested that participants had not actually played the game (i.e., when the velocity quickly decreased to 200 and did not rise above 250 for the remainder of the game).

This left 221 participants (93 female) whose data were retained for analysis. Ages ranged from 20 years to 65 years, with a mean age of 36.51 years (std. deviation 9.73). There were 82 participants in the control group, 68 in the DDA group, and 71 in the incremental group.

## 3 Results

The 14-item online version of the Flow Short Scale was shown to have acceptable reliability (Cronbach's  $\alpha = .834$ ).

Chi-square tests of homogeneity showed that the three groups did not significantly differ in terms of gender ( $\chi^2(3) = 1.4$ ,  $p = .497$ ), number of days per week spent playing video games ( $\chi^2(3) = 23.348$ ,  $p = .055$ ), or age ( $\chi^2(3) = 69.525$ ,  $p = .902$ ).

### 3.1 Significant results: performance

To analyse group differences in terms of overall mean score (i.e., participants' mean score over 19 levels of play), we ran a Kruskal-Wallis H test. The distributions of overall mean score were not similar for all groups. The DDA group had a smaller range (2.89) and smaller interquartile range (.83) than both the control group (range = 7.32, interquartile range = 3.01) and incremental group (range = 7.74, interquartile range = 2.89) (see Figure 2). The median values increased from the incremental group (3.74), to the control group (4.61) to the DDA group (5.05). These differences in median values were statistically significantly different between the groups,  $\chi^2(3) = 16.148$ ,  $p < .0005$ . Pairwise comparisons were then performed using Dunn's (1964) procedure with a Bonferroni correction for multiple comparisons. This analysis revealed statistically significant differences (adjusted p-values presented) in median values between the DDA group (mean rank = 135.31) and control

## DYNAMIC DIFFICULTY ADJUSTMENT IN VIDEO GAMES

group (mean rank = 106.91) ( $p = .02$ ), and between the DDA group and incremental group (mean rank = 92.44) ( $p < .0005$ ) groups, but not between the control group and the incremental group ( $p = .488$ ).

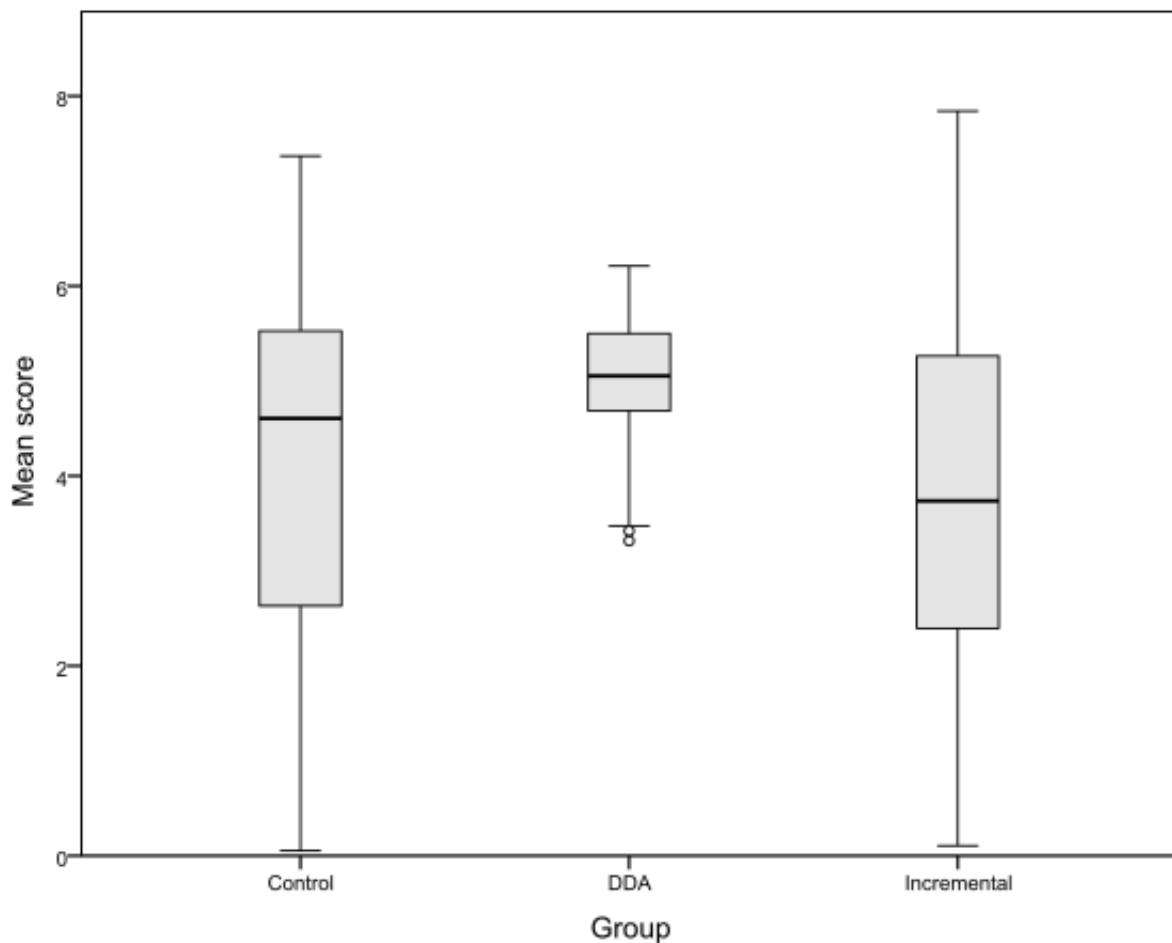


Figure 2. Boxplots of overall score for each of three groups, Control, Dynamic Difficulty Adjustment (DDA), and Incremental.

We analysed the differences in scores between the three groups further by conducting one-way Welch ANOVAs on mean score per level (i.e., mean score for each group for each of 19 levels of play). A small number of outliers in the DDA group and Incremental group were not removed. Levene's test for equality of variances showed that the assumption of homogeneity of variances was violated in levels 4 – 19 (p-values ranging from .012 to <.0005). Scores were significantly different between the groups during levels 1, 3, 5, and levels 9 – 19. Games-Howell post hoc analyses revealed that the DDA group had higher mean scores than the control group in levels 3 – 19; these differences were significant in levels 3, 5, and 9 – 19 (p-values ranging from .019 to <.0005). The DDA group also had significantly higher mean scores than the incremental group in levels 10 – 19 (p-values ranging from .013 to <.0005). The control group had significantly higher mean scores than the incremental group in levels 14 – 19 (p-values ranging from .014 to <.0005). These results are shown in Figures 3.1, 3.2, and 3.3.

## DYNAMIC DIFFICULTY ADJUSTMENT IN VIDEO GAMES

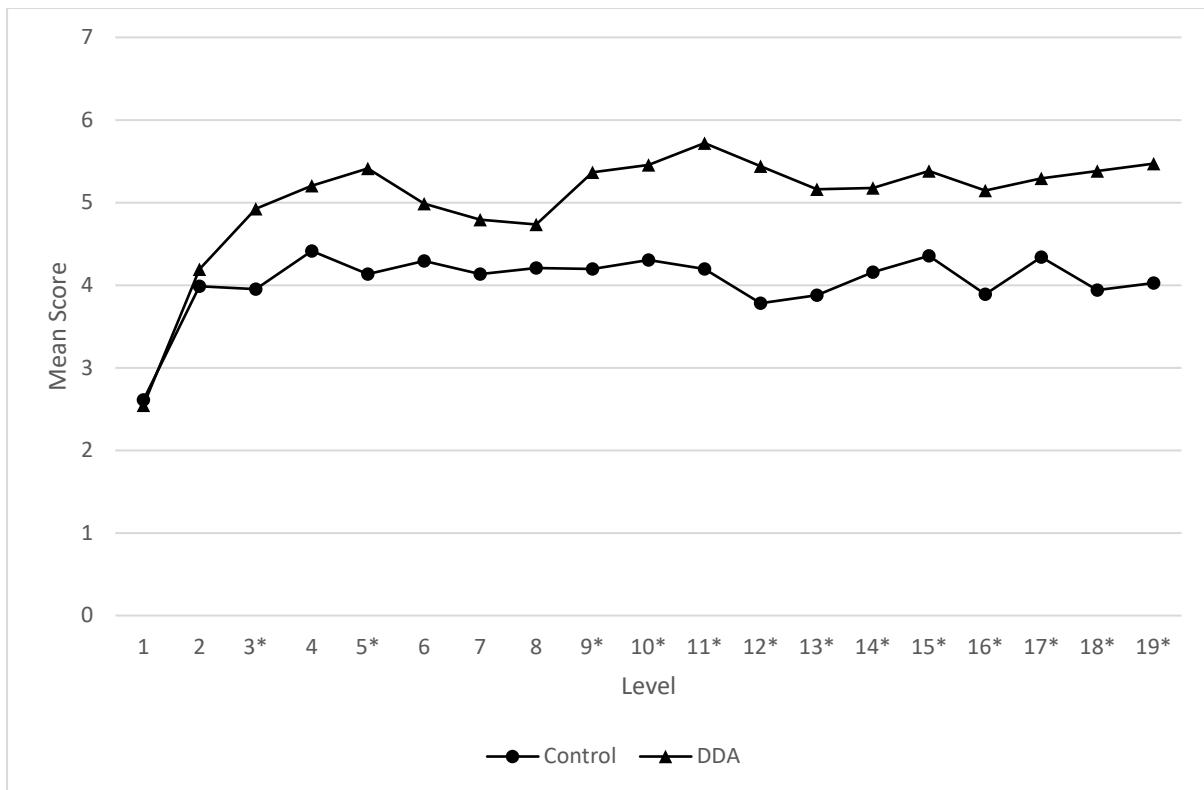


Figure 3.1. A comparison of mean score per level for the Control and Dynamic Difficulty Adjustment (DDA) groups. Significant differences are marked with a \*.

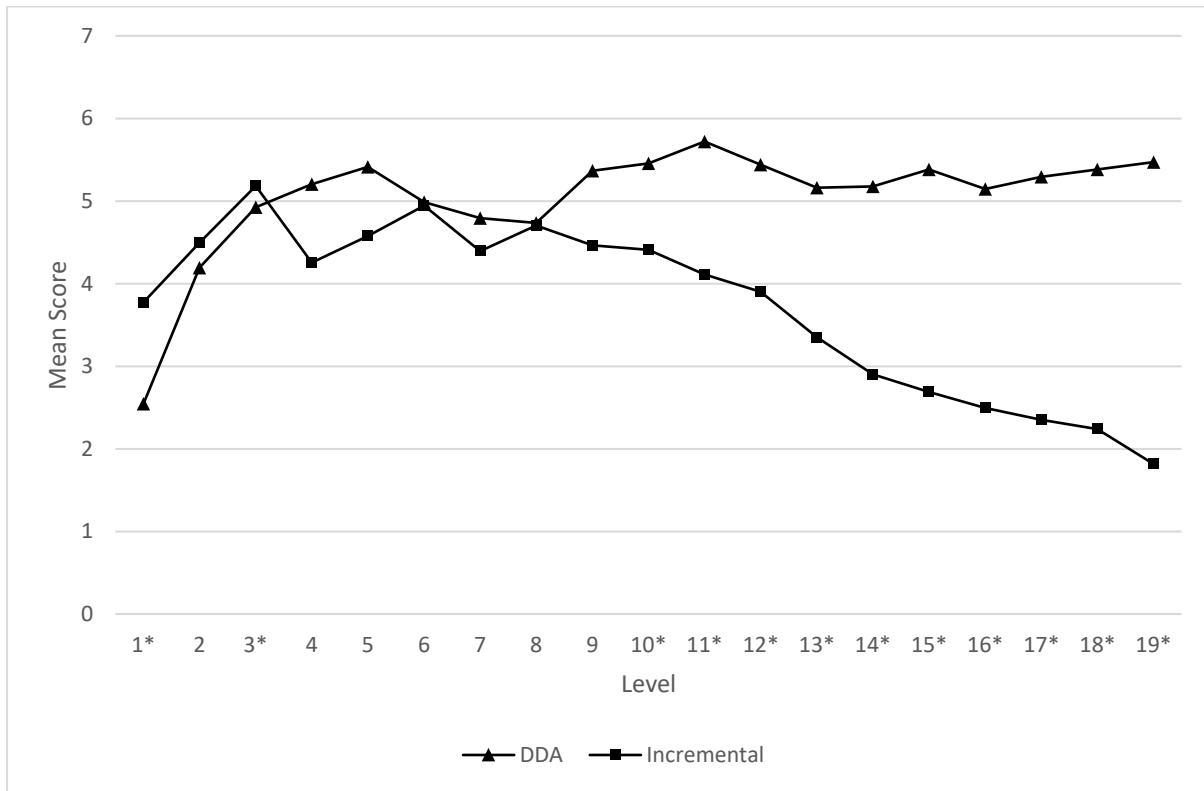


Figure 3.2. A comparison of mean score per level for the Dynamic Difficulty Adjustment (DDA) and Incremental groups. Significant differences are marked with a \*.

## DYNAMIC DIFFICULTY ADJUSTMENT IN VIDEO GAMES

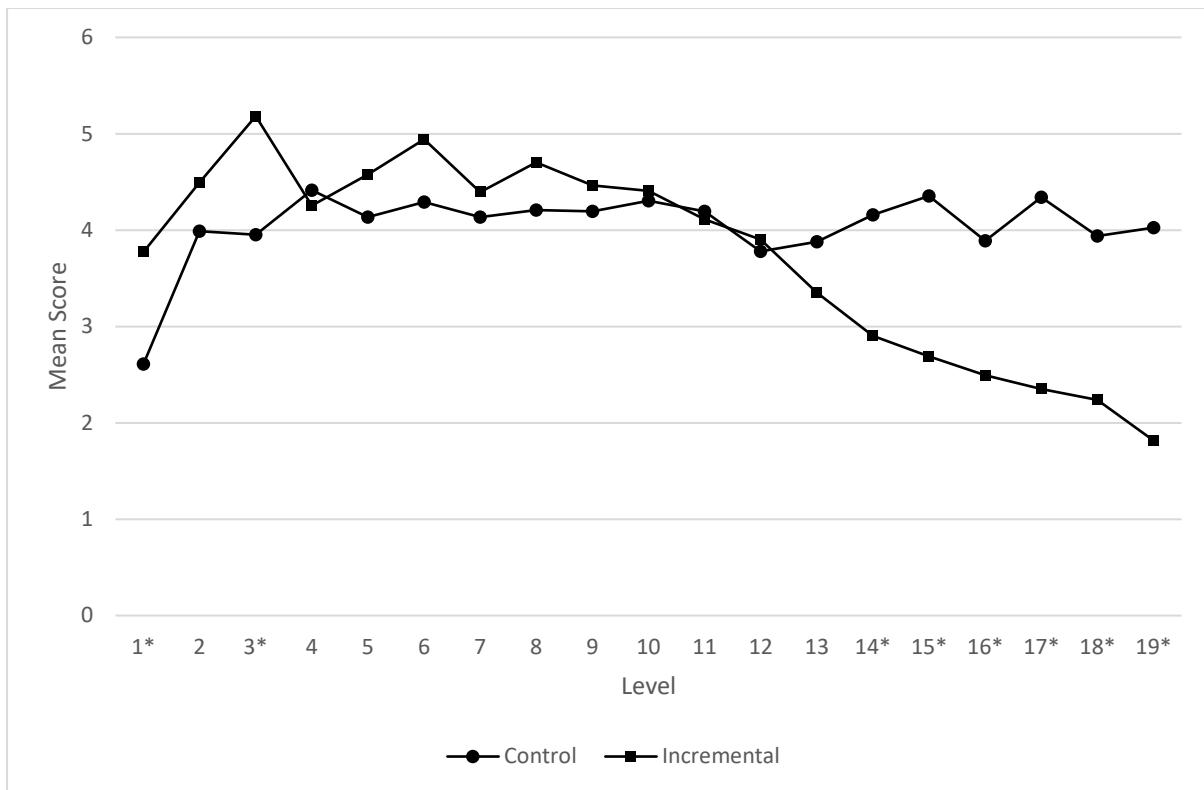


Figure 3.3. A comparison of mean score per level for the Control and Incremental groups. Significant differences are marked with a \*.

For the DDA group, we analysed the velocity values, which indicates the difficulty of the game (higher velocity is more difficult). First, we considered the mean velocity across participants at each measurement point (5 seconds between each measurement, 171 measurements considered). Figure 4 shows how velocity ranged across participants over time. We considered the relationship between each participant's (DDA group only) overall mean velocity over 855 seconds of gameplay, and each participant's overall mean score across 19 levels, using Pearson's correlation test. The data were linear, overall mean score was normally distributed ( $p > .05$ ), while overall mean velocity was not normally distributed ( $p < .0005$ ). There was a strong positive correlation between overall mean score and overall velocity,  $r(68) = .554$ ,  $p < .0005$ , with overall mean velocity explaining 31% of the variation in overall mean score

## DYNAMIC DIFFICULTY ADJUSTMENT IN VIDEO GAMES

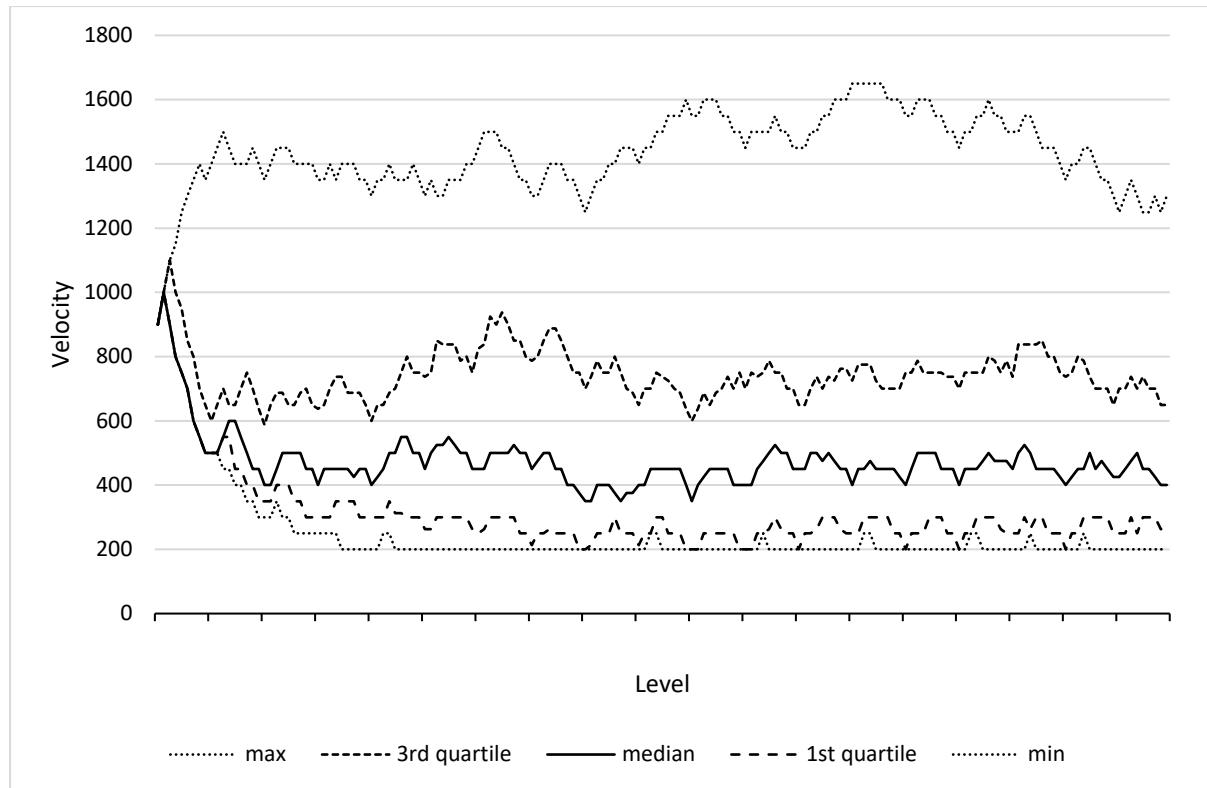


Figure 4. Range of meteor velocity per level for the Dynamic Difficulty Adjustment (DDA) group over 19 levels. As a higher velocity makes the game more difficult, and higher velocities are only achieved by players who perform better, velocity here can be used as an index of player performance.

### 3.2 Non-significant results: flow, anxiety, challenge, and enjoyment

We also ran Kruskal-Wallis H tests to analyse differences between the groups in terms of the 3 factors of the Flow Short Scale, and the single-item for Enjoyment.

In the case of Flow, the distributions were not similar for all groups (assessed by visual inspection of a box plot). The median values increased from the incremental group (4.9), to the DDA group (4.95), to the control group (5.3). These differences were not statistically significantly different between the groups,  $\chi^2(3) = 4.087$ ,  $p = .130$ .

In the case of Anxiety, the distributions were similar for all groups. The median values increased from the incremental group (3.67), to the control group (4.0) and DDA group (4.0). These differences were not statistically significantly different,  $\chi^2(3) = 3.336$ ,  $p = .189$ .

In the case of Challenge, the distributions were not similar for all groups and the median values were the same for all three groups (4.0).

In the case of Enjoyment, the distributions were not similar for all groups. The median values increased from the incremental group (5.0), to the control group (6.0) and the DDA group (6.0). These differences were not statistically significantly different,  $\chi^2(3) = 3.628$ ,  $p = .163$ .

### 3.3 Correlations

We also considered relationships between flow, anxiety, challenge, enjoyment, and overall mean score across all three groups. Flow was positively correlated with Anxiety ( $r(221) = .462$ ,  $p < .0005$ ), Challenge ( $r(221) = .476$ ,  $p < .0005$ ), and Enjoyment ( $r(221) = .675$ ,  $p < .0005$ ), Anxiety was positively correlated with Challenge ( $r(221) = .531$ ,  $p < .0005$ ) and

Enjoyment ( $r(221) = .511, p < .0005$ ), and Challenge was positively correlated with Enjoyment ( $r(221) = .501, p < .0005$ ). Overall mean score was significantly negatively correlated with Challenge ( $r(221) = -.181, p = .007$ ).

### 3.4 Hypotheses

In the case of hypothesis (1) – that DDA will produce greater player performance than either incremental or no difficulty adjustment – we are able to reject the null hypothesis.

However, for hypotheses (2) and (3), we are unable to reject the nulls. That is, we cannot reject the hypotheses that there is no difference between DDA, incremental difficulty adjustment, and the control group in terms of either player experience of flow or player enjoyment.

## 4 Discussion, limitations, and future work

The research presented here investigated dynamic difficulty adjustment in a video game, with the adjustment based on feedback about the player's performance, and using a machine learning algorithm to select the appropriate setting for a single variable which directly affected game difficulty. We found that this type of difficulty adjustment led to greater overall player performance over approximately 15 minutes of play, than either incremental difficulty adjustment or a no-adjustment control group. In addition, the range of performance in the DDA group was smaller than the other groups, and overall performance in the DDA group correlated with overall mean difficulty across 15 minutes of play.

In line with previous research, our results suggest that performance-based DDA is suitable for reducing skill differentials between players. This provides further evidence of the suitability of this relatively simple approach to DDA in facilitating competitive and/or collaborative play between players with different skill levels. We believe that this approach could therefore be used to increase the accessibility of video games for people with disabilities.

It is also interesting to note that the players in the DDA group showed a wide range of abilities, as indexed by the range of difficulty settings recorded during the experiment (Figure 4). However, the overall mean performance of this group increased more than both the control and incremental groups, with significant differences in mean performance in most levels of the game between the DDA and other groups. It is therefore possible, in line with the results of Sampayo-Vargas et al. (2013), that DDA provides a scaffold to players of a range of abilities, by making the game easier when their performance drops; this scaffold is then removed when their performance increases. However, there may be detrimental effects associated with DDA if players are aware that it is operating (Baldwin et al., 2014; Baldwin et al., 2016), and further research should investigate how players' awareness of DDA is related to their experience of playing a video game.

Our results show no significant differences between the three conditions on all the self-reported measures of player experience (flow, enjoyment, challenge, and anxiety). Previous research on performance-based DDA has found mixed results on self-report measures of player experience such as enjoyment, engagement, motivation, and challenge. There are several possible explanations for this. Firstly, it may be the case that performance-based approaches to DDA have less or no effect on self-reported player experience than affective

approaches. This is feasible, as several studies have found that player perceptions of gameplay experience are related to factors other than player skill or performance, such as gameplay experience (Alexander et al., 2013), motivations, personality, or preferences (Karpinskyj et al., 2014). If the aim of DDA is to increase players' positive perception of the gameplay experience, then affective approaches to DDA may be more useful. Secondly, within performance-based DDA approaches, there is a large range of factors which could influence player experience. In this study, we used simple measures of player performance to alter a single variable affecting game difficulty. There are many other factors we could have chosen. In addition, we could have provided a different range of difficulty settings (e.g., with larger or smaller increments), used a different target success rate, adapted the difficulty more or less frequently, and so on. These adjustments could lead to different results, and future research should consider, not just the difference between adaptive and non-adaptive difficulty adjustment, but also differences between alternative approaches to DDA. Thirdly, as discussed in Section 1.4, self-report data obtained from MTurk may be less reliable than data obtained from traditional sources. While we included attention-check items to identify participants who were potentially selecting random answers to the questions, found high reliability for our questionnaire, and found expected correlations between flow, anxiety, challenge, and enjoyment, we did not use any other techniques to identify participants who were not engaged with the questionnaire. For example, it has been suggested that MTurk data can be made more reliable by explicitly asking participants if they answered the questions genuinely and assuring them that they will still be paid if they admit they did not (Rouse, 2015). Note that this limitation is somewhat mitigated in this study as we removed responses from participants whose performance data indicated that they had not engaged with the game. However, it is still feasible that some participants engaged with the game and read the questions but still provided unreliable data simply by not providing considered answers.

While this study focused on a performance-based approach to DDA, it is important to recall that the POSM algorithm used is not specific to this context or to the game used in our experiment. This means that our approach, in principle, could be matched with any other feedback that can be quantified. It could therefore be used, not only with other performance measures, but, crucially, with affective measures, as any real-time quantitative data could be used as the feedback upon which difficulty is updated. This could include, for example, heart rate, galvanic skin response, or neurological activity. In future work, it would be informative to test the feasibility of the POSM algorithm in affective DDA systems.

## 5 Conclusion

This study makes several contributions to research on dynamic difficulty adjustment in video games. Our results show that performance-based dynamic difficulty adjustment, based on the Partially ordered set master (POSM) algorithm (Missura & Gärtner, 2011) can be used to increase player performance and reduce performance differentials in a 2D video game. We also demonstrate the feasibility of conducting video games research using an experimental game via Amazon Mechanical Turk. We make recommendations for future research to further investigate the effects of dynamic difficulty adjustment on enjoyment and experience of flow (for which we found non-significant results), such as using different feedback measures (including affective feedback), adjusting different game variables, and

## DYNAMIC DIFFICULTY ADJUSTMENT IN VIDEO GAMES

implementing additional steps to ensure the reliability of data gathered by player self-report.

### References

- Adams, E. (2008, May 14). The designer's notebook: Difficulty modes and dynamic difficulty adjustment. Retrieved from [https://www.gamasutra.com/view/feature/132061/the\\_designers\\_notebook\\_.php](https://www.gamasutra.com/view/feature/132061/the_designers_notebook_.php)
- Alexander, J. T., Sear, J., & Oikonomou, A. (2013). An investigation of the effects of game difficulty on player enjoyment. *Entertainment Computing*, 4(1), 53-62.
- Altimira, D., Clarke, J., Lee, G., Billinghurst, M., & Bartneck, C. (2017). Enhancing player engagement through game balancing in digitally augmented physical games. *International Journal of Human-Computer Studies*, 103, 35-47.
- Ang, D. & Mitchell, A. (2017). Comparing effects of dynamic difficulty adjustment systems on video game experience. In Proceedings of the Annual Symposium on Computer-Human Interaction in Play, 317-327. ACM.
- Baldwin, A., Johnson, D., & Wyeth, P. A. (2014). The effect of multiplayer dynamic difficulty adjustment on the player experience of video games. In *Proceedings of the extended abstracts of the 32nd annual ACM conference on Human factors in computing systems* (pp. 1489-1494). ACM.
- Baldwin, A., Johnson, D., & Wyeth, P. (2016). Crowd-pleaser: Player perspectives of multiplayer dynamic difficulty adjustment in video games. In *Proceedings of the 2016 Annual Symposium on Computer-Human Interaction in Play* (pp. 326-337). ACM.
- Barrett, N., Swain, I., Gatzidis, C., & Mecheraoui, C. (2016). The use and effect of video game design theory in the creation of game-based systems for upper limb stroke rehabilitation. *Journal of Rehabilitation and Assistive Technologies Engineering*, 3, 2055668316643644.
- Bateman, S., Mandryk, R. L., Stach, T., & Gutwin, C. (2011). Target assistance for subtly balancing competitive play. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 2355-2364). ACM.
- Bevilacqua, F., Backlund, P., & Engstrom, H. (2015). Proposal for non-contact analysis of multimodal inputs to measure stress level in serious games. In *Games and Virtual Worlds for Serious Applications (VS-Games), 2015 7th International Conference on*. IEEE.
- Bevilacqua, F., Backlund, P., & Engstrom, H. (2016). Variations of Facial Actions While Playing Games with Inducing Boredom and Stress. In *Games and Virtual Worlds for Serious Applications (VS-Games), 2016 8th International Conference on*. IEEE.
- Booth, M. (2009). The AI systems of Left 4 Dead. Keynote presented at the 5<sup>th</sup> Artificial Intelligence and Interactive Digital Entertainment Conference, Stanford, CA.
- Bontchev, B. (2016). Adaptation in Affective Video Games: A Literature Review. *Cybernetics and Information Technologies*, 16(3), 3-34.
- Casler, K., Bickel, L., & Hackett, E. (2013). Separate but equal? A comparison of participants and data gathered via Amazon's MTurk, social media, and face-to-face behavioral testing. *Computers in Human Behavior*, 29(6), 2156-2160.

## DYNAMIC DIFFICULTY ADJUSTMENT IN VIDEO GAMES

- Chang, D. M. J., (2013). Dynamic difficulty adjustment in computer games. Retrieved from <http://studylib.net/doc/8266212/dynamic-difficulty-adjustment-in-computer-games>
- Chen, J. (2007). Flow in games (and everything else). *Communications of the ACM*, 50(4), 31-34.
- Cheung, J. H., Burns, D. K., Sinclair, R. R., & Sliter, M. (2017). Amazon Mechanical Turk in organizational psychology: An evaluation and practical recommendations. *Journal of Business and Psychology*, 32(4), 347-361.
- Csikszentmihalyi, M. (1975). Beyond boredom and anxiety: Experiencing flow in work and play. San Francisco: Jossey-Bass.
- Csikszentmihalyi, M. (1988). The flow experience and its significance for human psychology. In M. Csikszentmihalyi & I. Csikszentmihalyi (Eds.), *Optimal experience: Psychological studies of flow in consciousness* (pp. 15–35). Cambridge: Cambridge University Press.
- Dunn, O. J. (1964). Multiple comparisons using rank sums. *Technometrics*, 6(3), 241-252.
- Engeser, S. (Ed.). (2012). *Advances in flow research*. Springer Science & Business Media.
- Entertainment Software Association (2017). Essential facts about the computer and video game industry. [http://www.theesa.com/wp-content/uploads/2017/09/EF2017\\_Design\\_FinalDigital.pdf](http://www.theesa.com/wp-content/uploads/2017/09/EF2017_Design_FinalDigital.pdf)
- Fleischer, A., Mead, A. D., & Huang, J. (2015). Inattentive responding in MTurk and other online samples. *Industrial and Organizational Psychology*, 8(2), 196-202.
- Gerling, K. M., Miller, M., Mandryk, R. L., Birk, M. V., & Smeddinck, J. D. (2014). Effects of balancing for physical abilities on player performance, experience and self-esteem in exergames. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems* (pp. 2201-2210). ACM.
- Hernandez, H. A., Ye, Z., Graham, T. C., Fehlings, D., & Switzer, L. (2013). Designing action-based exergames for children with cerebral palsy. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 1261-1270). ACM.
- Hocine, N., Gouaïch, A., Cerri, S. A., Mottet, D., Froger, J., & Laffont, I. (2015). Adaptation in serious games for upper-limb rehabilitation: an approach to improve training outcomes. *User Modeling and User-Adapted Interaction*, 25(1), 65-98.
- Hunicke, R. (2005). The case for dynamic difficulty adjustment in games. In *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology* (pp. 429-433). ACM.
- Hwang, S., Schneider, A. L. J., Clarke, D., Macintosh, A., Switzer, L., Fehlings, D., & Graham, T. C. (2017). How Game Balancing Affects Play: Player Adaptation in an Exergame for Children with Cerebral Palsy. In *Proceedings of the 2017 Conference on Designing Interactive Systems* (pp. 699-710). ACM.
- Ilici, L., Wang, J., Missura, O., & Gärtner, T. (2012). Dynamic difficulty for checkers and Chinese chess. In *Computational Intelligence and Games (CIG), 2012 IEEE Conference on* (pp. 55-62). IEEE.

## DYNAMIC DIFFICULTY ADJUSTMENT IN VIDEO GAMES

- Jackson, S. A., & Marsh, H. W. (1996). Development and validation of a scale to measure optimal experience: The Flow State Scale. *Journal of sport and exercise psychology, 18*(1), 17-35.
- Jensen, M. M., & Grønbæk, K. (2016). Design strategies for balancing exertion games: A study of three approaches. In *Proceedings of the 2016 ACM Conference on Designing Interactive Systems* (pp. 936-946). ACM.
- Karpinskyj, S., Zambetta, F., & Cavedon, L. (2014). Video game personalisation techniques: A comprehensive survey. *Entertainment Computing, 5*(4), 211-218.
- Leiker, A. M., Bruzi, A. T., Miller, M. W., Nelson, M., Wegman, R., & Lohse, K. R. (2016). The effects of autonomous difficulty selection on engagement, motivation, and learning in a motion-controlled video game task. *Human movement science, 49*, 326-335.
- Mason, W., & Suri, S. (2012). Conducting behavioral research on Amazon's Mechanical Turk. *Behavior research methods, 44*(1), 1-23.
- Missura, O., & Gärtner, T. (2011). Predicting dynamic difficulty. In *Advances in Neural Information Processing Systems* (pp. 2007-2015).
- Nagle, A., Novak, D., Wolf, P., & Riener, R. (2014). The effect of different difficulty adaptation strategies on enjoyment and performance in a serious game for memory training. In *Serious Games and Applications for Health (SeGAH), 2014 IEEE 3rd International Conference on* (pp. 1-8). IEEE.
- Orvis, K. A., Horn, D. B., & Belanich, J. (2008). The roles of task difficulty and prior videogame experience on performance and motivation in instructional videogames. *Computers in Human behavior, 24*(5), 2415-2433.
- Paolacci, G., & Chandler, J. (2014). Inside the Turk: Understanding Mechanical Turk as a participant pool. *Current Directions in Psychological Science, 23*(3), 184-188.
- Perttula, A., Kiili, K., Lindstedt, A., & Tuomi, P. (2017). Flow experience in game based learning—a systematic literature review. *International Journal of Serious Games, 4*(1).
- Rheinberg, F., Vollmeyer, R., Engeser, S. (2003). Die Erfassung des Flow-Erlebens [Measuring flow experiences]. In J. Stiensmeier-Pelster, F. Rheinberg (Eds.), *Diagnostik von Motivation und Selbstkonzept. Tests und Trends, Vol. 2*, Hogrefe, Göttingen (2003), pp. 261-279
- Robb, N., Waller, A., & Woodcock, K. A. (2019). Developing a task switching training game for children with a rare genetic syndrome linked to intellectual disability. *Simulation & Gaming, 50*(2), 160-179.
- Rouse, S. V. (2015). A reliability analysis of Mechanical Turk data. *Computers in Human Behavior, 43*, 304-307.
- Sampayo-Vargas, S., Cope, C. J., He, Z., & Byrne, G. J. (2013). The effectiveness of adaptive difficulty adjustments on students' motivation and learning in an educational computer game. *Computers & Education, 69*, 452-462. doi: 10.1016/j.compedu.2013.07.004
- Shaker, N., Yannakakis, G., & Togelius, J. (2010). Towards automatic personalized content generation for platform games. In 6th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, AIIDE 2010.

## DYNAMIC DIFFICULTY ADJUSTMENT IN VIDEO GAMES

- Sharek, D., & Wiebe, E. (2015). Investigating Real-time Predictors of Engagement: Implications for Adaptive Videogames and Online Training. *International Journal of Gaming and Computer-Mediated Simulations (IJGCMS)*, 7(1), 20-37.  
doi:10.4018/IJGCMS.2015010102
- Sherry, J. L. (2004). Flow and media enjoyment. *Communication Theory*, 14(4), 328–347.
- Thibault, R. T., Lifshitz, M., & Raz, A. (2016). The self-regulating brain and neurofeedback: experimental science and clinical promise. *Cortex*, 74, 247-261.
- Weibel, D., Wissmath, B., Habegger, S., Steiner, Y., & Groner, R. (2008). Playing online games against computer-vs. human-controlled opponents: Effects on presence, flow, and enjoyment. *Computers in Human Behavior*, 24(5), 2274-2291. doi: 10.1016/j.chb.2007.11.002
- Williams, D., Yee, N., & Caplan, S. E. (2008). Who plays, how much, and why? Debunking the stereotypical gamer profile. *Journal of Computer-Mediated Communication*, 13(4), 993-1018.
- Xue, S., Wu, M., Kolen, J., Aghdaie, N., & Zaman, K. A. (2017). Dynamic Difficulty Adjustment for Maximized Engagement in Digital Games. In *Proceedings of the 26th International Conference on World Wide Web Companion* (pp. 465-471). International World Wide Web Conferences Steering Committee.

# A Temporal Data-Driven Player Model for Dynamic Difficulty Adjustment

Alexander E. Zook and Mark O. Riedl

School of Interactive Computing, College of Computing  
Georgia Institute of Technology  
[{a.zook,riedl}@gatech.edu](mailto:{a.zook,riedl}@gatech.edu)

## Abstract

Many computer games of all genres pit the player against a succession of increasingly difficult challenges such as combat with computer-controlled enemies and puzzles. Part of the fun of computer games is to master the skills necessary to complete the game. Challenge tailoring is the problem of matching the difficulty of skill-based events over the course of a game to a specific player's abilities. We present a tensor factorization approach to predicting player performance in skill-based computer games. Our tensor factorization approach is data-driven and can predict changes in players' skill mastery over time, allowing more accurate tailoring of challenges. We demonstrate the efficacy and scalability of tensor factorization models through an empirical study of human players in a simple role-playing combat game. We further find a significant correlation between these performance ratings and player subjective experiences of difficulty and discuss ways our model can be used to optimize player enjoyment.

## Introduction

Many computer games of all genres pit the player against a succession of increasingly difficult challenges: combat with NPC enemies, puzzles, strategic planning and execution, etc. The player is expected to master a set of skills pertaining to the game mechanic over the course of the game. Some believe that this mastery of the game is a fundamental aspect to having fun in computer games (Koster 2005). However, contemporary computer games are being played by increasingly diverse audiences that differ in their skills and interests in games. This increasing variability in ability, speed of mastery, and growing diversity in tastes for game aesthetic and narrative content has prompted a recent growth of interest in automated methods to fit game content to these diverse abilities and interests. These efforts require both modeling the abilities and interests of players as well as adapting existing game content to those differences.

Many games revolve around *skill-based events*, periods of game play, such as combat or puzzles, in which the player must perform a specific skill. We see two main challenges to adapting computer games to fit individual player differences: *challenge tailoring* and *challenge contextualization*.

Challenge tailoring (CT) is the problem of matching the difficulty of skill-based events over the course of a game to a specific player's abilities. For example, in an action role-playing game such as *The Legend of Zelda* challenge tailoring may manifest as configuring the number, health, or damage dealt by various enemies at various times throughout the game. CT is similar to *Dynamic Difficulty Adjustment* (DDA), which only applies to online, real-time changes to game mechanics to balance difficulty. In contrast, CT generalizes DDA to both online and offline optimization of game content and is not limited to adapting game difficulty. Challenge contextualization (CC) is the related problem of constructing game events that set up the conditions for skill events and motivate their occurrence to the player. For example, the challenge of slaying a dragon may be contextualized by the dragon kidnapping a princess. Challenge contextualization includes common AI problems of quest generation, story generation in games, and interactive storytelling.

In this paper, we focus on the challenge tailoring problem in adventure role-playing games. Realizing challenge tailoring requires both a player model and an algorithm to adapt content based on that model. Effective player modeling for the purposes of challenge tailoring requires a data-driven approach that is able to predict player behavior in situations that may have never been observed. Because players are expected to master skills over time when playing a game, the player model must also account for temporal changes in player behavior, rather than assume the player remains fixed. Modeling the temporal dynamics of a player enables an adaptive game to more effectively forecast future player behavior, accommodate those changes, and better direct players toward content they are expected to enjoy. Further, forecasting enables player models to account for interrelations among sequences of experiences—accounting for how foreshadowing may set up a better future revelation or how encountering one set of challenges builds player abilities to overcome related challenges that build off of those.

In this paper we present and evaluate a temporal player modeling approach. We employ tensor factorization techniques to create temporal models of objective player game performance over time in a turn-based role-playing game and demonstrate the efficacy of our approach over related data-driven methods through comparisons from an empirical study. We model performance instead of difficulty be-

cause performance is objectively measurable while difficulty is subjective; we show difficulty and performance are significantly correlated for this particular domain. Finally, we suggest how our tensor factorization player model may be used for challenge contextualization.

## Related Work

Smith et al. (2011) overview the landscape of player modeling in computer games. In their taxonomy, we are investigating individual, empirical, generative player models. Research in player modeling has typically addressed the challenge tailoring problem either by developing purely behavioral models or relying on predictions that ignore temporal changes in player data. Hunicke and Chapman (2004) model players by computing the average and variance of player damage and item inventory. Dynamic Difficulty Adjustment is achieved via a hand-crafted policy prescribing actions to take based on player health and inventory states. Magerko et al. (2006) interactive story players using a vector of competence levels for various skills and associated confidence values. The system selects from a pool of challenges based on a best fit between the characteristics of the challenge event and the current state of the skill vector. van Lankveld et al. (2008) role-playing game players using their progress and health, dynamically adjusting sets of presented enemies to enforce a given level of player health over progress. In contrast, our data-driven modeling approach explicitly forecasts changes in player performance, combines information across players, and proactively constructs a long-term set of challenges based on these predictions.

Subjective self-report indications of challenge have also been used to dynamically tailor game play (Yannakakis and Togelius 2011). Pedersen et al. (2009) train a neural network to predict player self-reported experiences of fun, challenge, and frustration based on measures of player behavior and in-game content. Yannakakis, Lund, and Hallam (2006) employ a neural network to predict player self-reported interest. Our approach extends these models by correlating time-varying measures of performance to self-report measures, enabling forecasts of player experience forward in time.

While we believe our work is the first application of tensor factorization to challenge tailoring problems, we note that similar techniques have been used to model student performance over time on standardized tests (Thai-Nghe, Horvath, and Schmidt-Thieme 2011).

## Player Model

We explore tensor factorization techniques for modeling player performance in action role-playing games. While we focus on action role-playing games, we believe our techniques generalize to any games that make regular use of skill-based events. Tensor factorization techniques decompose multidimensional measurements into latent components that capture underlying features of the high-dimensional data. Tensors generalize matrices, moving from the two-dimensional structure of a matrix to a three or more dimensional structure. For our player modeling approach we extend two-dimensional matrices representing player perfor-

mance against particular enemy types to add a third dimension representing the time of that performance measure.

We chose to use tensor factorization due to its favorable scaling properties, ability to cope with missing data, high accuracy, speed in generating predictions, and previous success in other applications. Tensor factorization is an extension of matrix factorization, which has been used in collaborative filtering applications—such as the Netflix Prize data mining competition—to great success. Matrix factorization offers the key advantage of leveraging information from a group of users that has experienced a set of content to make predictions for what a new group of individuals that has only been partially exposed to that content will do. Specifically, user data is represented in a  $M = U \times I$  matrix indicating user preference ratings on items and decomposition extracts latent factors relating to users and items. Tensor factorization adds more dimensions, such as time, and extracts latent factors related to these other dimensions as well. Matrix and tensor factorization scale effectively to large numbers of users and items, handle missing information from users and achieve high accuracy (Koren and Bell 2011; Su and Khoshgoftaar 2009).

Formally, we represent player data in a tensor  $Z = U \times I \times T$ . In our simple spell-casting action role-playing game (described in the next section),  $U$  is the player (“user”),  $I$  is the spell type (“item”) and  $T$  is the time of the performance recording. CP decomposition is a generalization of singular value decomposition from matrices to tensors. In CP decomposition the three-dimensional  $Z$  tensor is decomposed into a weighted combination of three vector latent factors,

$$Z \approx \sum_{k=1}^K \lambda_k w_k \circ h_k \circ q_k$$

where  $\circ$  is the vector outer product,  $\lambda_k$  are positive weights on the factors,  $w_k$  are player factors,  $h_k$  are spell type factors, and  $q_k$  are time factors.  $K$  is the number of components used for each factor as an approximation of the true structure, keeping the set of the  $K$  most important components found (Kolda and Bader 2009). The decomposition can be computed by minimizing the root mean squared error between the factor inner product above and true data values, iteratively fitting each of the factors while fixing values of the other factors until convergence. We employ the N-way toolbox (Andersson and Bro 2000) to perform this decomposition. Predictions in this model consist of taking the inner product of these three factors, a computationally efficient process.

## Experiment

We focus on skill-based aspects of games that require procedural knowledge: the ability to correctly act in given circumstances, typically with limited time for decision-making. To test our tensor factorization model we conducted a study of player learning in a turn-based role-playing game. Below we present the game used for the study, study methodology, and the results of comparing our modeling technique to related approaches.

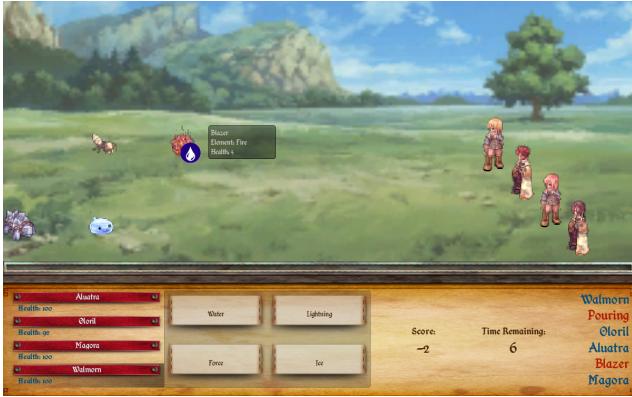


Figure 1: A battle between monsters and the player team.

Table 1: Spell Effectiveness Matrix.

| Attack ↓ Def. → | fire | water | acid | ice | light. | earth | force | undeath |
|-----------------|------|-------|------|-----|--------|-------|-------|---------|
| fire            | 1    | 0     | 1    | 2   | 1      | 2     | 1     | 0       |
| water           | 2    | 1     | 0    | 1   | 0      | 1     | 2     | 1       |
| acid            | 1    | 2     | 1    | 0   | 1      | 0     | 1     | 2       |
| ice             | 0    | 1     | 2    | 1   | 2      | 1     | 0     | 1       |
| lightning       | 1    | 2     | 1    | 0   | 1      | 0     | 1     | 2       |
| earth           | 0    | 1     | 2    | 1   | 2      | 1     | 0     | 1       |
| force           | 1    | 0     | 1    | 2   | 1      | 2     | 1     | 0       |
| undeath         | 2    | 1     | 0    | 1   | 0      | 1     | 2     | 1       |

## Game Domain

We implemented a turn-based role-playing game in which the player leads a group of four characters through a sequence of spell-casting battles against groups of enemy monsters. For the purpose of this study, we ignore the quest-like contextualization of the battles. See Figure 1 for the game’s battle interface. Turns are limited to 10 seconds to require mastery of a complex spell system.

Each enemy is associated with one of eight spell types and player-controlled characters can attack with four of the eight possible spells. Casting a particular spell against an enemy of a particular type results in an attack being effective, ineffective, or super-effective, resulting in normal damage, no damage, or double damage against an enemy (see Table 1).

We intentionally created a spell system that was difficult to completely memorize, but contained intuitive combinations—water spells are super-effective against fire enemies—and unintuitive combinations—undead spells are super-effective against force enemies—ensuring that skill mastery could only be achieved by playing the game. Note that pairs of spells—e.g., fire and force—are repeated in Table 1. This ensures a simpler underlying skill structure for players to learn; there are effectively only four spells. A scoring system based on spell effectiveness motivates players to learn; effective spells earn two points, ineffective spells earn zero points, and super-effective spells earn five points. Enemy attacks decrease player score by one. Player characters were assigned different spell sets, forcing players to learn the spell system.

## Study Methodology

We recruited 32 participants to play our role-playing game, which was simplified to present a series of battles one after another, as in Figure 1. In our study we first gave players five minutes to review a text document explaining the spell system and the game interface. Once familiar with the game, players completed the sequence of eleven battles while we recorded the performance of the player in terms of effectiveness of spells chosen against enemies. After each battle we additionally asked players to report how difficult and enjoyable the battle was on a 5-point Likert scale.

We recorded each spell players cast on each enemy on each turn and battle in the game along with the associated performance value: 0 for ineffective, 1 for effective, and 2 for super-effective. Because spell effectiveness determines damage to enemies, player behavior traces varied in the number of turns taken, but had the same number of battles. We average performance for each spell type across all turns within a given battle, leaving the performance value missing for any spells not used in the battle.

We hypothesize that a tensor factorization approach will outperform non-tensor approaches. The tensor factorization model organizes player data in a three-dimensional tensor (player, enemy spell type, battle number), where each point in the tensor is the average performance of the player in attacking foes of a given spell type during a given battle. Spell types not used in a given battle were recorded as missing values. This produces a  $32 \times 8 \times 11$  tensor for our 32 players, 8 spell types, and 11 battles. We compared the tensor model to two other models: matrix factorization using an unfolded tensor, and matrix factorization averaging over time. The matrix unfolding of this tensor simply concatenates battles column-wise, producing a  $32 \times 88$  matrix recording player performance on each spell type and battle number combination. The final matrix factorization approach averaged player performance against spell types across all battles, removing any temporal information. Data points represent average player performance over their entire battle history against an enemy type. If our hypothesis is confirmed, then the tensor model captures additional structure lost in the unfolded matrix model when all time points are concatenated together.

We also hypothesize that there is an inverse relationship between objective, measurable player performance and subjective, self-reported difficulty. Should this hypothesis hold it will verify that in the context of action role-playing games we can use skill performance as a proxy for difficulty.

## Results

We first compared players according to a variety of collected demographic information (age, gender, race, ethnicity, prior video game and role-playing game experience) and found no significant differences among these groups in our data set, validating the use of a single model across all players. We measured the 10-fold cross-validated *percent variance explained* of the tensor, matrix unfolded, and time-averaged models. Percent variance explained is an error measure that compares the total variation in the data across cases with the variation remaining after a model has been applied to the

data. It describes how well a model captures variations in the data, with higher percentages indicating more powerful explanations.

The tensor model outperforms the other techniques for three or more components (see Table 2). While the tensor model does not achieve substantially better performance than the unfolded or time-averaged models on few components it shows much greater performance on larger numbers of components, reflecting its greater capacity to model complex underlying structure in the data. Notably, the tensor model shows comparable high performance for three, four, or five factors, indicating the spell matrix reflects an underlying structure testing the corresponding number of skills. As noted earlier our spell matrix actually only contains four spells (each spell has two names). This intuitively explains a result suggesting four underlying factors—the four unique spells—with similar numbers of factors reflecting a moderate amount of noise around this underlying information. The 100% score for the time-averaged model with 8 factors reflects the fact that this model is modeling 8 spell types with 8 factors and thus can trivially recover the same model.

| components | tensor | unfolded | timeavg |
|------------|--------|----------|---------|
| 2          | 92.59  | 90.82    | 95.86   |
| 3          | 92.18  | 89.44    | 72.41   |
| 4          | 88.72  | 86.30    | 39.99   |
| 5          | 90.11  | 61.77    | 45.02   |
| 6          | 89.11  | 63.66    | 24.28   |
| 7          | 87.11  | 24.29    | 36.30   |
| 8          | 80.05  | -10.81   | 100.00  |

Table 2: Comparison of percent variance explained by different models using varying numbers of factors.

We evaluated the scaling of the tensor model in three ways: (1) how well the model scales with additional data; (2) how well the model forecasts into the future for players; and (3) how well the model scales in the number of players used by the system. In all three cases we use 10-fold cross-validated percent variance explained averaged over 10 repetitions of the experiment. Our first assessment hid a random subset of all our data and generated predictions for those missing values as shown in Figure 2. Model performance increased with the amount of total data used, with the simple two-component model showing high performance across all amounts of data while the more complex six- and eight-component models required approximately 60% of the data to achieve high performance. Thus, tensor factorization techniques perform well with this kind and amount of data, with more complex models requiring additional data but achieving high performance with more information.

Our second assessment evaluated the accuracy of the tensor model predictions on future events by hiding all future battles for a group of target players (Figure 3). The simple two-component tensor model achieved high performance regardless of the number of battles used, while the more complex six- and eight-component models required approximately seven battles to achieve comparable performance.

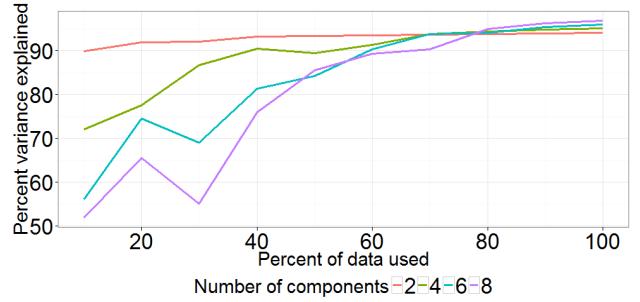


Figure 2: Percent variance explained of the tensor model as a function of the amount of data randomly subsampled from the data set, averaged over 10 repetitions.

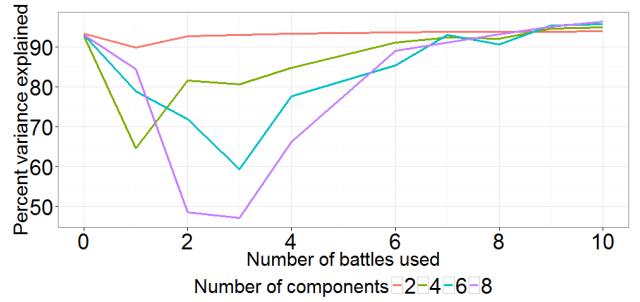


Figure 3: Percent variance explained of the tensor model when using first subset of battle data from randomly selected players, averaged over 10 repetitions.

Thus, tensor factorization models can generate useful forecasts after only a small number of battles observed for any given player. From a game design perspective, this suggests a relatively short training level in which the player can demonstrate skills can yield reasonable predictive power.

Our third assessment examined the tensor model accuracy when training and testing on varying numbers of players. We randomly subsampled from our full data set to use 6, 11, 16, 21, or 27 players and performed the same cross-validation analysis as above. Accuracy improved on the 2 component model from 81.17% variance explained with 6 players to 86.52% with 27 players, while the 4 component model improved from 52.55% to 86.29%, respectively. Other numbers of components showed similar trends, converging to approximately 86% accuracy with 27 players. The results of these three assessments demonstrate the power of the tensor factorization technique to scale with additional players and additional data for those players.

Finally, we found a significant correlation between objectively measure performance and subjectively experienced difficulty. Performance levels significantly differed (Kruskal-Wallis test for analysis of variance,  $p < 0.001$ ) between subjective difficulty ratings. A Kruskal-Wallis test assesses whether responses (here performance) in different groups (difficulty ratings) shows significant differences—it is a non-parametric alternative to the standard analysis of variance tests that assume the data has a Gaussian distri-

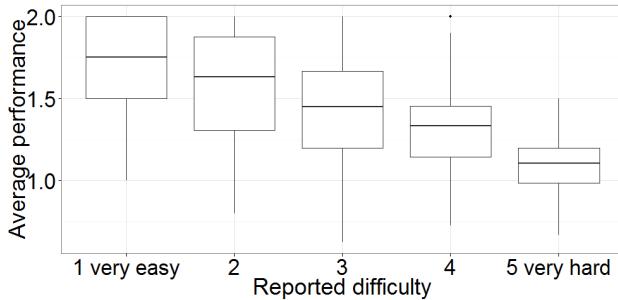


Figure 4: Comparison between objective performance values for subjective difficulty ratings, showing median values and the range from 25<sup>th</sup> and 75<sup>th</sup> percentiles.

bution, as our data showed skewing in performance measures for different response groups. A Dunnett’s test found all adjacent pairs of difficulty ratings significantly differed ( $p < 0.05$ ) in performance value, ranging from  $-0.11$  to  $-0.22$  (Figure 4). Dunnett’s test applies a more powerful version of the Student’s t-test to performance multiple comparisons among groups. Thus our second hypothesis is confirmed; objective measures of skill performance are inversely related to perceived difficulty.

## Content Adaptation

Our temporal player model can predict player performance on skill-based events in the future. To perform challenge tailoring of a game, the player model must be combined with information about how to use the model. In the next sections we describe (1) a target expected performance to compare predictions of an individual user’s performance against, and (2) an algorithm to select and/or parameterize skill-based events to bring predicted player performance in line with target performance.

## Target Performance Curve

We expand upon the concept of a *performance curve* (Zook et al. 2012), as an indication of the desired player performance over a sequence of skill-based challenges. In our game this indicates desired player performance across a sequence of battles, provided by the human designer. Figure 5 shows an example of a performance curve. This particular curve seeks a decrease in performance over time until the very end of the game. This game should appear to a player to increase in difficulty from challenge to challenge, even as the player continues to master the skills involved. Other curves are possible as well. A curve expressed by  $p = c$  (a horizontal line at a fixed constant,  $c$ ) indicates a game in which the difficulty appears to remain the same, even as the player’s skills improve. That is, the challenges may be parameterized to be more difficult, but because the challenge level is increasing at the same rate as player skill mastery, there should be little or no perceived change in difficulty. More complicated patterns, such as a series of rises and falls, can express complex designer intentions.

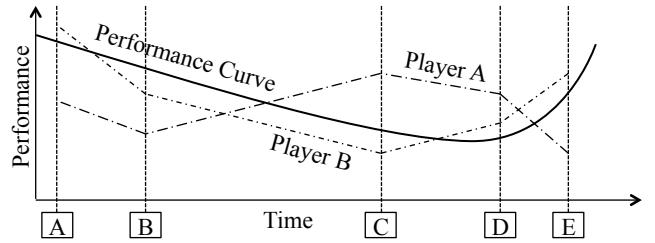


Figure 5: An example performance curve indicating a game that gets progressively more difficult until the very end, even as the player’s skills improve. Dotted lines indicated predicted performance for two players.

Note that performance curves are specifications of desired player performance, rather than difficulty. Although our studies show that performance and difficulty are inversely correlated, there may be other aspects of the game—such as ambiguity in game information—that are not necessarily captured in the performance metric. A performance curve bypasses the ambiguity of subjective difficulty while enabling designers to specify how they desire a particular performance metric to vary over the course of an experience.

## Challenge Tailoring and Contextualization

Given a performance curve and a temporal player model, challenge tailoring is the problem of selecting and spacing a sequence of skill-based events—in our case, battles—that match the predicted performance over time to the desired performance. This is a discrete optimization problem—battles are discrete units and the goal is to minimize the difference between predicted and desired performance values for these battles. A variety of techniques may be applied to solve these problems including constraint satisfaction, dynamic programming, and heuristic search techniques such as genetic algorithms (Smith and Mateas 2011; Togelius et al. 2011; Sorenson, Pasquier, and DiPaola 2011; Zook et al. 2012). In contrast to the reactive, near-term changes typically employed in DDA (Magerko, Stensrud, and Holt 2006; Hunnicke and Chapman 2004), temporal player models are able to also proactively restructure long-term content to optimize a global player experience.

Challenge contextualization is the problem of selecting non-skill-based events that explain, motivate, or justify the skill-based events. Quest generation and narrative generation techniques may be used to provide this contextualization of skill-based events, although other, non-narrative contextualization may exist as well. Challenge tailoring and challenge contextualization may be performed in a pipelined fashion or in parallel. Pipeline techniques afford modular and compositional approaches to tailoring game content. In particular, if tailoring of skill-based events takes precedence over contextualization such that contextualization (i.e., the narrative, quest, or mission) can be sub-optimal then one might choose to solve the discrete optimization problem of selecting and spacing all skill-based events before “filling the gaps” with non-skill-based, contextualizing events (cf., (Magerko, Stensrud, and Holt 2006)). Fixing the skill-based

events prior to contextualization makes challenge tailoring easier, but constrains the searchable context space.

Parallel techniques, conversely, may explore the full space of joint skill- and non-skill combinations, but trade this flexibility for greater complexity in the tailoring task. For example, we describe a technique that searches for a complete sequence of skill- and non-skill-based events that simultaneously optimizes for player performance and context (Zook et al. 2012). The question of whether to use a pipeline versus parallel approach depends on the importance of contextualization relative to skill tailoring and the extent to which skill- and non-skill-based events appear seamless.

## Conclusions

As computer games grow in popularity personalization of game content—which we cast as the paired problems of challenge tailoring and challenge contextualization—will also grow in importance. Unlike many other player modeling approaches to challenge tailoring and dynamic difficulty adjustment, we use a data driven approach that explicitly incorporates a temporal component. This temporal component allows us to more accurately forecast *future* player performance by modeling changes in a player’s skills.

A performance curve provides authorial control over the game in the form of a target level of performance. Since objective performance is inversely correlated with subjective difficulty, a performance curve can guide an algorithm in the selection and parameterization of skill-based events over time. The tensor factorization model gives a system insight into how the player will perform on various sequences of challenges. With these tools, there are many discrete-optimization algorithms that can solve the challenge tailoring problem. Challenge contextualization, though outside the scope of this paper, can further ensure challenges make sense to the player while promoting player skill mastery.

Mastery of skills is correlated with fun (Koster 2005). A game that is able to tailor its difficulty to meet the abilities of the player may be perceived as more enjoyable to a wider range of players because it is never unintentionally boringly easy or frustratingly hard. This in turn has the potential to increase game replayability and make certain games accessible to a wider diversity of players.

## Acknowledgments

The project or effort described here has been sponsored by the U.S. Army Research, Development, and Engineering Command (RDECOM). Statements and opinions expressed do not necessarily reflect the position or the policy of the United States Government, and no official endorsement should be inferred.

## References

- Andersson, C., and Bro, R. 2000. The N-way toolbox for MATLAB. *Chemometrics and Intelligent Laboratory Systems* 52(1):1–4.
- Hunicke, R., and Chapman, V. 2004. AI for dynamic difficulty adjustment in games. In *Proceedings of the AAAI Workshop on Challenges in Game Artificial Intelligence*.
- Kolda, T., and Bader, B. 2009. Tensor decompositions and applications. *SIAM review* 51(3):455–500.
- Koren, Y., and Bell, R. 2011. Advances in Collaborative Filtering. In Ricci, F.; Rokach, L.; Shapira, B.; and Kantor, P. B., eds., *Recommender Systems Handbook*. Boston, MA: Springer. 145–186.
- Koster, R. 2005. *A Theory of Fun in Game Design*. Paraglyph press.
- Magerko, B.; Stensrud, B.; and Holt, L. 2006. Bringing the schoolhouse inside the box - a tool for engaging, individualized training. In *Proceedings of the 25th Army Science Conference*.
- Pedersen, C.; Togelius, J.; and Yannakakis, G. N. 2009. Modeling Player Experience in Super Mario Bros. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games*.
- Smith, A., and Mateas, M. 2011. Answer set programming for procedural content generation: A design space approach. *IEEE Transactions on Computational Intelligence and AI in Games* 3(3):187–200.
- Smith, A.; Lewis, C.; Hullett, K.; Smith, G.; and Sullivan, A. 2011. An inclusive view of player modeling. In *Proceedings of the 6th International Conference on Foundations of Digital Games*.
- Sorenson, N.; Pasquier, P.; and DiPaola, S. 2011. A Generic Approach to Challenge Modeling for the Procedural Creation of Video Game Levels. *IEEE Transactions on Computational Intelligence and AI in Games* 3(3):229–244.
- Su, X., and Khoshgoftaar, T. M. 2009. A Survey of Collaborative Filtering Techniques. *Advances in Artificial Intelligence* 2009:1–19.
- Thai-Nghe, N.; Horvath, T.; and Schmidt-Thieme, L. 2011. Factorization Models for Forecasting Student Performance. In *Proceedings of the 4th International Conference on Educational Data Mining*.
- Togelius, J.; Yannakakis, G.; Stanley, K.; and Browne, C. 2011. Search-Based Procedural Content Generation: A Taxonomy and Survey. *IEEE Transactions on Computational Intelligence and AI in Games* 3(3):172–186.
- van Lankveld, G.; Spronck, P.; and Rauterberg, M. 2008. Difficulty scaling through incongruity. In *Proceedings of the 4th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*.
- Yannakakis, G. N., and Togelius, J. 2011. Experience-Driven Procedural Content Generation. *IEEE Transactions on Affective Computing* 2:147–161.
- Yannakakis, G.; Lund, H.; and Hallam, J. 2006. Modeling children’s entertainment in the playware playground. In *Proceedings of the 2nd IEEE Symposium on Computational Intelligence and Games*.
- Zook, A.; Riedl, M. O.; Holden, H. K.; Sotilare, R. A.; and Brawner, K. W. 2012. Automated scenario generation: Toward tailored and optimized military training in virtual environments. In *Proceedings of the 7th International Conference on the Foundations of Digital Games*.

# Dynamic Difficulty Adjustment in Tetris\*

Diana Lora, Antonio A. Sánchez-Ruiz, Pedro A. González-Calero and Marco A. Gómez-Martín

Departamento de Ingeniería del Software e Inteligencia Artificial

Universidad Complutense de Madrid, Spain

dlora@ucm.es, antsanch@fdi.ucm.es, pedro@fdi.ucm.es, marcoa@fdi.ucm.es

## Abstract

A game fun to play is the one that provides challenges to the players corresponding to their skills. Most of the games have different preconfigured difficulty levels, but they do not adjust the difficulty dynamically to the player skill. In this work, we explore the idea of creating clusters from previous game traces to capture different playing styles in Tetris and then use those clusters to decide how much help the system should provide to new players giving them good Tetris pieces. In our experiments players report improvements in terms of game experience.

## Introduction

The main goal of any game is to entertain its users. According to the players psychological model describe by Daniel Cook (Cook 2007), players are driven by the desire of mastering new skills. Fun is achieved once the players overcome a challenge and masters a new skill. After that, the game gives rewards for the hard work and creates new challenges to conquer. The game creates a loop of learning-mastery-reward which needs to be balance in order to keep players interested.

A game fun to play is the one that provides challenges to the players corresponding to their skills. The difficulty is considered a subjective factor which is derived from the interaction between the player and the proposed challenge. This is not a static property, because it changes depending on the time spent by the player mastering a skill (Missura and Gärtner 2009; Hunicke 2005). With Dynamic Difficulty Adjustment (DDA) is possible to maintain the right balance depending on the player's skills. However, DDA usually involves a great amount of work for game programmers and designers that cannot be reused in other video games. In this context, research is important to decrease the costs related to development of adaptive games using different techniques such us, for example, automatically extracting information from traces of previous games. For the purpose of this paper, a game's trace describes the evolution of different variables during the game and it may contain data of the interaction

between the user and the game as well as data about the virtual environment where user interacts.

In this paper we present our approach to DDA in Tetris, a very popular video game. We extract traces from previous games and build a case base in which each case describes how the user places a sequence of consecutive pieces in the game board. Then we use clustering to group the cases according to the skill level of the player. When a player starts a new game we look at his first movements to find the most similar cluster. Then the system provides dynamic help to the player choosing "good" Tetris pieces from time to time. In our experiments, using DDA users obtain higher scores and report improvements in terms of their game experience.

The rest of the paper is organized as follows. First, we discuss the features extracted from the game traces to characterize the style of play, and the process to create different clusters to group games depending on the player's skill level. Then, we explain our proposed approach to provide help and dynamically adjust the difficulty of the game. Next, we analyze the results obtained in some experiments with new players when we apply DDA. The paper closes with related work, conclusions and directions for future research.

## Tetris and Feature Selection

Tetris (Figure 1) is a very popular video game in which the player has to place different tetromino pieces that fall from the top of the screen in a rectangular game board. When a row of the game board is filled, i.e. it has no holes, the row disappears and all pieces above dropped one row. The pieces fall faster and faster as the game progresses until the board is full and the game is over (Breukelaar et al. 2004).

We use an implementation of the game called TetrisAnalytics that looks like an ordinary Tetris from the point of view of the player but provides extra functionality to extract, store and reproduce game traces. From these traces we can extract the most significant features, determine the skills of the player, and dynamically adjust the difficulty of the game to improve user experience. In order to build the training set, we collected game traces from 15 different players with diverse skills levels and they played around 300 games.

Each time a new piece appears on the top of the game board, the player has to make two different decisions. The first one, that we call *tactical*, is to decide the final location and rotation of the piece. The second decision involves how

\* Supported by Spanish Ministry of Economy and Competitiveness under grant TIN2014-55006-R  
Copyright © 2015, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

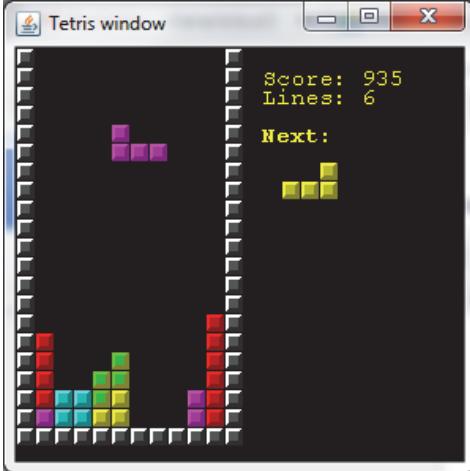


Figure 1: A screen of the Tetris game.

to *move* the piece to its final location (e.g. to rotate the piece twice and move it to the left 3 times). In this work we restrict our study to tactical decisions and how they define the skill level of the player. However, we think that we could also extract valuable information from the concrete movements (speed, cadence, movements undone, ...) and we expect to extend our work in the future.

For each tactical decision (i.e. for each piece in the game) we extract the following features:

- The current and next type of piece.
- The final location (row, column and rotation) of the piece.
- The state of each cell (free / occupied) in the highest 4 occupied rows of the game board. We only store those rows because there is a high chance that the player will place the current piece in that region.
- The points obtained by placing the current piece.
- The maximal points the player could have obtained if she would have performed the best possible action (according to a heuristic and greedy AI player that reduces the height of the board and the number of holes).
- The current score, number of completed lines and speed of the game.
- The height of the board (as the highest occupied row in the board).

## Classifying Players Depending on their Skill Level

Unfortunately one single tactical decision is not enough to decide the skill level of the player with confidence, we need to consider longer sequences of pieces. The length of these sequences is an important factor because the more pieces we consider the better we can describe the style of play, but we will also have to wait longer to detect the skill level of a new player and adjust the difficulty of the game.

After some experiments we concluded that 10 tactical decisions (or pieces) is a good compromise. In order to select

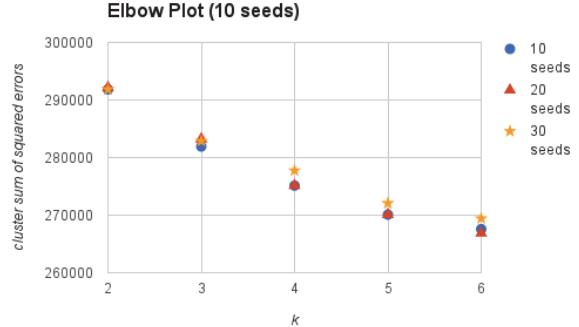


Figure 2: Elbow plot with the mean squared error as a function of the number of clusters.

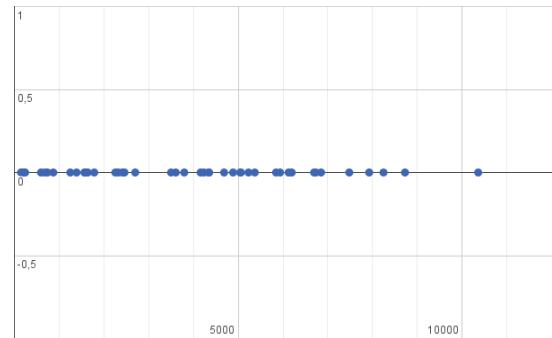


Figure 3: Total score of each game on the x axis.

this value we identified in the game traces a few players with similar scores but very different playing styles and then we performed several clustering algorithms joining the features of  $n$  consecutive pieces and increasing the number  $n$ . With  $n = 10$  those players were classified in the same cluster so we concluded that we should wait at least 10 pieces before trying to predict the skill level of a new player.

From the game traces we created a case base in which each case stores the features of 10 consecutive tactical decisions. Each case represents a small fragment of the game and partially captures the player's playing style. Our next goal is to find clusters in the case base to identify different groups of players and then to use those clusters to predict the skill level of new players.

In order to select an appropriate number of clusters we used the same technique as Drachen et. al (Drachen et al. 2012a), using the k-means algorithm and varying  $k$  from 2 to 6. Figure 2 shows an elbow plot with the mean squared error as a function of the number of clusters. The first clusters add much information (explain a lot of variance) but then the marginal gain drops. Figure 3 shows the total score of each game on the x axis. We can see in some regions a lot of points followed by another region with no point or just a few ones. From the two graphs, we can conclude that a good option for a simple game like Tetris is to establish three clusters that we will label as *newbie*, *average* and *expert*.

Figure 4 shows the scores of the cases assigned to each

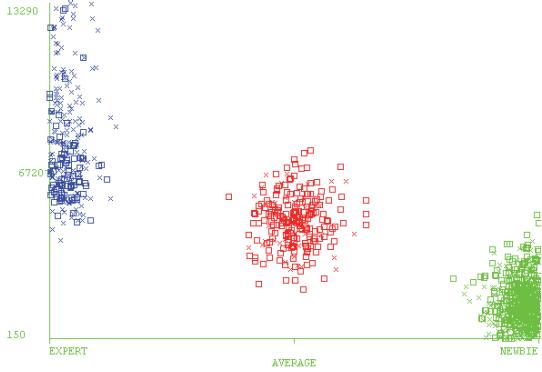


Figure 4: Total Score Vs. Clusters

| Cluster | Cases | Characteristics   |
|---------|-------|---|
| NEWBIE  | 34%   | Homogeneous distribution of pieces along the board.<br>Tendency to locate the pieces at the right and left sides of the board.<br>Low rotation rate.  |
| AVERAGE | 17%   | The distribution of majority of pieces along the board are in the lower half (rows 10 to 19).<br>There is no preference to locate the pieces breadways of the board.<br>High rotation rate. |
| EXPERT  | 48%   | The distribution of majority of pieces along the board are in the lower half (rows 13 to 19).<br>There is no preference to locate the pieces breadways of the board.<br>High rotation rate. |

Table 1: Clusters, size and interpreted playing styles.

cluster. We can see that each cluster is related to a group of players with a different skill level. Table 1 shows the percentage of cases in each cluster and some descriptions about the different playing styles that are more representative in each cluster. For example, *newbie* players tend to place the pieces more often on the left and right sides of the board than in the middle, and they rotate the pieces less than more experienced players. *Average* and *expert* players, on the other hand, play most of the time placing pieces in the lower half of the game board and only when the game is close to the end and the speed of the falling pieces is very high, they are forced to place the pieces in the upper area.

### Dynamic Difficulty Adjustment

To do dynamically difficulty adjustment in a game, first we need to predict the skill level of the new player from the location of the first pieces in the game. As we explained in the previous section, in order to make a prediction with some level of confidence we need to wait until the player has placed at least 10 pieces in the game board. We also re-

| Newbie | Average | Expert |
|--------|---------|--------|
| 50%    | 30%     | 10%    |

Table 2: How often the system helps the user giving her a good Tetris piece.

quire the height of the board (the highest occupied row) to be over 1/3 of the total height. When both requirements have been fulfilled, we have at least one sequence of 10 pieces (probably more) to predict the player's skill level.

The prediction is made by classifying the sequence of pieces in one of the three clusters. If there are several sequences of pieces available, the player's skill level is decided by a majority vote. This way, the skill level is decided based on the similarity between the current partial game and the beginnings of the games stored in the training set.

After we know the player's skill level (newbie, average or expert), we have to decide when and how to help them. The difficulty in Tetris depends mainly on two parameters: the type of pieces and the falling speed. In our work we decided to focus only on the former parameter because the new pieces that appear in the game are supposed to be random, and it is very difficult for a player to realize they are not random if we pick them carefully. The other parameter, the speed at which the pieces fall, is supposed to depend only on the number of lines cleared in the game, and we think it is easier for a player that plays several games to perceive changes in that parameter.

Another important decision to make is related to how often we help the user. Table 2 shows in how many pieces the system provides a *good* next piece. Obviously, the lower the level of the player the more help the system provides.

When the system decides to help the player, it checks how good each type of piece is in the current game board according to a heuristic function and selects one of the best three pieces randomly. For each type of piece we evaluate every board that can be obtained by placing the piece in every position and rotation, and then each type of piece is ranked according to its best final board. We use a very simple heuristic function that computes the number of empty rows in the board and subtracts the number of *holes* in the rows with pieces.

It is interesting to note that our first approach was to provided always the best piece but it turned out it was not a good idea because some pieces like the square or the stick are good quite often and they appeared too many times. It was notorious that the system was cheating and the players perceived it negatively.

## Experiments and Results

We asked 16 different users to play 4 games of Tetris and then evaluate their game experience using a 5-point Likert scale. We provide help using DDA only in games number 2 and 4, while games 1 and 3 were normal Tetris games to compare the results. Of course, the users did not know in which games DDA was active.

Figure 5 shows the average score among all the players in each game. We can see that the players obtained higher

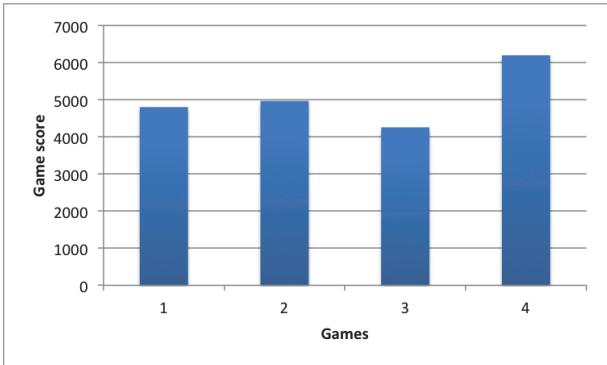


Figure 5: Average score in each game. DDA was active only in games 2 and 4.

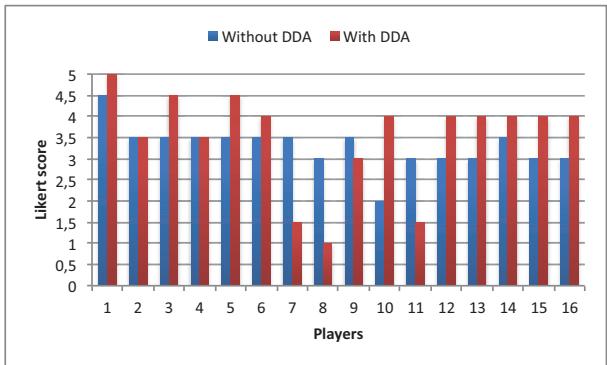


Figure 6: Game experience evaluation with and without DDA.

scores in games 2 and 4 in which DDA was active. Although we do not show it, these results are consistent with the duration of the games that was higher when DDA was active.

More interesting are the results shown in Figures 6 and 7 regarding the subjective evaluation of the game experience. Figure 6 shows the average satisfaction of each player in games with and without DDA active. In the *x* axis, we have the players and in the *y* axis the likert score they had given. For each player, the blue bar corresponds to the user experience in the two games DDA was not active. Whereas, the red bar is the user experience had when DDA was active and giving the user the “right” piece depending on their profile. We can see that, in general, the user satisfaction was higher with DDA active. Figure 7 shows the number of players that experienced DDA as a positive and negative effect. From 16 players, the game experience improved in 10 cases, did not affect in 2, and was worst in 4.

Although our results are preliminary, DDA seems to have a positive impact in terms of user satisfaction because the game adjust the difficulty according to the player’s skills level. This way, when the player needs help, the game gives them one of three best possible pieces to score more points that would have.

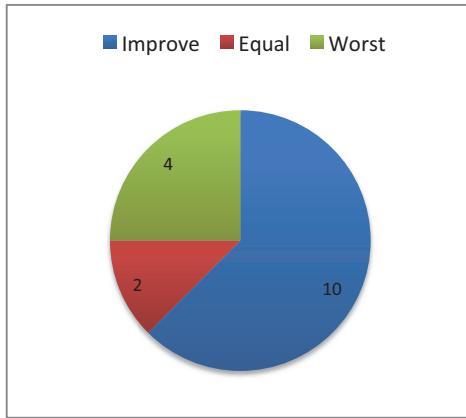


Figure 7: Number of users whose game experience improved with DDA.

## Related Work

In video games, behavior analysis is a novel issue compared to other fields of application. One popular way of behavioral analysis is *segmentation* or *categorization*. Categorization is used for any analysis technique aiming at reducing a number of users in a few descriptive profiles, regardless of the method applied (e.g. segmentation, clustering y classification) (Drachen et al. 2012b). Players profiling can be used for testing, improvement of game design, creation of new monetization strategies, game customization, among others (Drachen, Canossa, and Yannakakis 2009; Mahlman et al. 2010).

With unsupervised learning techniques is possible to extract pattern from behavioral data without knowing too much about them (Drachen et al. 2013; Springer 2010b). The methods focuses on the structure and relationships between data, i.e. they look for patterns among features. Clustering is the process of grouping a set of objects such a way that the elements belonging to the same cluster are similar to each other and different from those that are part of other groups. Additionally, clustering allows reducing the dimensionality of the dataset (Springer 2010a). One category of cluster algorithms is centroid based clustering, like k-means (Lloyd 1982) which is often used in players profiling because of its popularity (Han and Kamber 2006).

On top of that, an entertaining game keeps the player so engaged that he could lose track of time. To make this possible, it is important to customize the game to their particular skills. This way the game won’t present unintended consequences, that could frustrate the goals and aspirations of the user. For this reason, having a balanced difficulty level and consistency is very important in video games. Here is where becomes obvious that a game should provide challenges according to who is playing. With DDA is possible to modulate difficulty based on players interaction. In commercial games, the main objective is to make games fun and interesting in order to improve revenue. Whereas, in serious games helps to increase the learning level of the player (Missura and Gärtner 2009; Fields and Cotton 2011). One popular approach of DDA is *Rubber band AI* which cre-

ates a virtual link between the player and its enemies. This way, if the player “pulls” in one direction, i.e. if the user plays better or worse, then its enemies will be dragged in the same direction showing a simple or complex behavior (Missura and Gärtner 2009). Supervised learning techniques focuses on imitation, analysis and prediction of the behavior of the player; developing better opponents and dynamic adaptation of the difficulty of the game (King and Chen 2009; Missura and Gärtner 2009; Yannakakis and Hallam 2009).

## Conclusions

Fun games provide challenges to the players according to their skills. As the player masters their skills the game should adapt itself to provide new challenges. In this work we have presented our approach to Dynamic Difficulty Adjustment (DDA) in Tetris. First, we collect traces from players with different skill levels and extract some features describing how they place the pieces in the game board. Then we build cases gathering together sequences of consecutive pieces and capturing fragments of the games. Next we find clusters in the case base to represent different playing styles and relate them to 3 skill levels: newbie, average and expert. Based on the skill level of new players, we provide some help in the form of “good” next Tetris pieces. Our experiments show that DDA has a positive effect for most players in their game experience.

As part of the future work we would like to extend our study to consider not only tactical decisions but the specific movements of the pieces until they are placed in their final positions. Our intuition is that the speed and number of times an expert player presses the keyboard is probably quite different from that of a newbie player. We would also like to explore other ways to capture playing styles and provide personalized help. For example, some particular player can have problems with a specific type of Tetris piece or with some configurations of the game board. Now, our difficulty adjustment is made once at the beginning of the gameplay when the pieces reached 1/3 of the board, but our intention is to dynamically identify when a player is receiving too much help from the game and it is necessary to recalculate their profile. On top of that, we would like to use clustering of time series data in order to create players profiles. This way, it won’t be necessary to join several tactical decisions together to find chronological patterns in user behavior.

Finally, we would like to use our approach to other games and see at what extent we are able to personalize the difficulty using only traces from previous experiences. The use of data mining techniques in video games can help to minimize the cost and effort of the development team to implement DDA manually. Game traces contain valuable information to determine the profiles of new users and personalize the behavior of the game to each specific player. The selection of variables to characterize the player’s skill level and the decision of which variables to modify in order to make the game easier or harder for the user, depend on the nature of each particular game. However, the data mining techniques described in this paper to extract information during game play, create clusters and identify different types

of users are standard and can be used in any game. In summary, we strongly think that game traces contain valuable information regarding past experiences of other players and therefore they are a promising source of knowledge.

## References

- Breukelaar, R.; Demaine, E. D.; Hohenberger, S.; Hoogeboom, H. J.; Kosters, W. A.; and Liben-Nowell, D. 2004. Tetris is hard, even to approximate. *Int. J. Comput. Geometry Appl.* 14(1-2):41–68.
- Cook, D. 2007. The chemistry of game design. [http://www.gamasutra.com/view/feature/129948/the\\_chemistry\\_of\\_game\\_design.php](http://www.gamasutra.com/view/feature/129948/the_chemistry_of_game_design.php). Consultado: 2015-05-26.
- Drachen, A.; Sifa, R.; Bauckhage, C.; and Thurau, C. 2012a. Guns, swords and data: Clustering of player behavior in computer games in the wild. In *2012 IEEE Conference on Computational Intelligence and Games, CIG 2012, Granada, Spain, September 11-14, 2012*, 163–170.
- Drachen, A.; Sifa, R.; Bauckhage, C.; and Thurau, C. 2012b. Guns, swords and data: Clustering of player behavior in computer games in the wild. In *2012 IEEE Conference on Computational Intelligence and Games, CIG 2012, Granada, Spain, September 11-14, 2012*, 163–170.
- Drachen, A.; Thurau, C.; Togelius, J.; Yannakakis, G.; and Bauckhage, C. 2013. Game data mining. In *Game Analytics, Maximizing the Value of Player Data*. Springer. 205–253.
- Drachen, A.; Canossa, A.; and Yannakakis, G. 2009. Player modeling using self-organization in tomb raider: Underworld. In Proceedings of IEEE Computational Intelligence in Games (CIG) 2009 (Milan, Italy).
- Fields, T., and Cotton, B. 2011. *Social Game Design: Monetization Methods and Mechanics*. MK Pub.
- Han, J., and Kamber, M. 2006. *Data Mining: Concepts and techniques*. Morgan Kaufmann.
- Hunicke, R. 2005. The case for dynamic difficulty adjustment in games. In *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology*, 429–433. ACM.
- King, D., and Chen, S. 2009. Metrics for social games. Presentation at the social games summit 2009, game developers conference. San Francisco, CA.
- Lloyd, S. 1982. Least squares quantization in pcm. *IEEE Transactions on Information Theory* 28(2).
- Mahlman, T.; Drachen, A.; Canossa, A.; Togelius, J.; and Yannakakis, G. 2010. Predicting player behavior in tomb raider: Underworld. In Proceedings of the 2010 IEEE Conference on Computational Intelligence in Games.
- Missura, O., and Gärtner, T. 2009. Player modeling for intelligent difficulty adjustment. In *Discovery Science, 12th International Conference, DS 2009, Porto, Portugal, October 3-5, 2009*, 197–211.
- Springer. 2010a. Clustering. In Sammut, C., and Webb, G., eds., *Encyclopedia of Machine Learning*. Springer US. 180–180.
- Springer. 2010b. Unsupervised learning. In Sammut, C., and Webb, G., eds., *Encyclopedia of Machine Learning*. Springer US. 1009–1009.
- Yannakakis, G., and Hallam, J. 2009. Real-time game adaptation for optimizing player satisfaction. *IEEE Trans. Comput. Intellig. and AI in Games* 1(2):121–133.

# DEGREE PROJECT

L

## Dynamic Difficulty Adjustment & Procedural Content Generation in an Endless Runner

Simon Vidman

**Computer Game Programming, bachelor's level**  
**2018**

Luleå University of Technology  
Department of Computer Science, Electrical and Space Engineering



# **Abstract**

## **Dynamic Difficulty Adjustment & Procedural Content Generation in an Endless Runner.**

Games are irritating when they are too hard, and boring when they are too easy. Level Eight is a game development company which is, during this project, developing an endless runner. This thesis describes research and implementation of a dynamic difficulty adjustment system in Level Eights' endless runner by considering several conditions of the player. This was accomplished with the help of the game environment by implementing procedural content generation and combine it with the dynamic difficulty adjustment system.

# **Sammanfattning**

## **Dynamic Difficulty Adjustment & Procedural Content Generation in an Endless Runner.**

Spel är irriterande när de är för svåra, och tråkiga när de är för lätta. Level Eight är ett spelutvecklingsföretag som under detta projekt utvecklar ett ändlöst löparspel. Denna avhandling beskriver forskning och genomförande av ett dynamiskt svårighets justeringssystem i Level Eights ändlöst löparspel genom att överväga flera villkor för spelaren. Detta uppnåddes med hjälp av spelmiljön genom att implementera procedurell innehållsgenerering och kombinera den med det dynamiska svårighets justering systemet.

## Acknowledgments

I would like to start off by giving my thanks to everyone who assisted me along the way. Special thanks to my examiner Patrik Holmlund for answering questions and giving continuous feedback on this paper.

A big thank you also goes out to Level Eight for giving me a place to do my thesis work and especially Pontus Bengtsson for giving me advice and feedback on the project but also the rest of the team at Level Eight. Lastly, a small thanks go to Jocce Marklund for providing helpful discussions.

# Abbreviations and Terms

- **Branch** - Independent line of development, isolated work
- **C#** - Object-oriented programming language, C-sharp
- **DDA** - Dynamic Difficulty Adjustment
- **PCG** - Procedural Content Generator
- **Chunk** - A pre-built piece of a level
- **GUI** - Graphical User Interface
- **IP** - Intellectual Property
- **Unity** - Game engine
- **L8** - Level Eight

# Table of Content

|   |           |
|---|-----------|
| <b>1 Introduction</b>                                 | <b>1</b>  |
| 1.1 Goal and Purpose                                  | 1         |
| 1.2 Limitations                                       | 2         |
| 1.3 Level Eight                                       | 2         |
| 1.4 Background  | 3         |
| 1.4.1 Endless Runner                                  | 3         |
| 1.4.2 Unity3D   | 4         |
| 1.4.3 Atlassian - SourceTree                          | 4         |
| 1.4.4 Flow  | 4         |
| 1.5 Related Work                                      | 5         |
| 1.5.1 Borderlands                                     | 5         |
| 1.5.2 Crash Bandicoot                                 | 6         |
| 1.5.3 Infinite Mario Bros                             | 6         |
| 1.5.4 Left 4 Dead                                     | 6         |
| 1.5.5 Mario Kart                                      | 7         |
| 1.6 Method  | 7         |
| 1.7 Social, Ethical, and Environmental Considerations | 7         |
| 1.7.1 Social considerations                           | 7         |
| 1.7.2 Ethical considerations                          | 8         |
| 1.7.3 Environmental considerations                    | 8         |
| <b>2 Design and Implementation</b>                    | <b>9</b>  |
| 2.1 Dynamic Difficulty Adjustment                     | 9         |
| 2.2 Procedural Content Generation                     | 11        |
| 2.3 Implementing the System                           | 13        |
| <b>3 Results</b>                                      | <b>17</b> |
| 3.1 End Results                                       | 17        |
| <b>4 Discussion</b>                                   | <b>18</b> |
| 4.1 Future work                                       | 18        |
| <b>5 Conclusion</b>                                   | <b>20</b> |
| <b>6 References</b>                                   | <b>21</b> |

# List of Figures

|  |    |
|--|----|
| 1.1 Level Eight's logotype                                   | 2  |
| 1.2 Robbery Bob image  | 2  |
| 1.3 Screenshot of the endless runner                         | 3  |
| 1.4 Mental state in terms of challenge level and skill level | 5  |
| 1.5 Various weapon components that can be combined           | 6  |
| 2.1 Performance function                                     | 10 |
| 2.2 Penalty  | 10 |
| 2.3 Performance  | 10 |
| 2.4 Death penalty  | 11 |
| 2.5 Performance thresholds                                   | 11 |
| 2.6 Grass field with obstacles                               | 12 |
| 2.7 Empty grass field  | 12 |
| 2.8 XML chunk in notepad++                                   | 12 |
| 2.9 Pseudo-code for penalty implementation                   | 15 |
| 2.10 Pseudo-code for performance implementation              | 15 |
| 2.11 Pseudo-code for difficulty implementation               | 16 |

# 1 Introduction

Video games are frustrating when they are too hard and boring when they are too easy, but what makes a game fun and interesting? The nature of fun experiences in games has been the topic of both systematic inquiry and speculations [7]. There are many theories from psychology and game studies focusing on the experiences of playing a game and one of the work is Csikszentmihalyi's concept of flow [8]. Players show a unique learning curve for every game that they play, it thus becomes difficult to entertain each player with preset difficulty levels that are defined by designers [3].

There's also been a lot of discussion about whether games should adapt to the skills of players [25]. However, most current technique limit adaptation to parameter adjustment. But if the parameter adaptation is applied to procedural content generation, then new levels can be generated in real-time in response to a players skill [22]. A game that many people recognize and have dynamic difficulty adjustment is Pocket billiards. The more balls your enemy pockets in, the better are your chances to hit your own balls on your next shot. So the game naturally becomes easier for the losing player and more difficult for the winning player because the scoring game pieces are also obstacles to the opponent [9].

## 1.1 Goal and Purpose

Implementing DDA with PCG in an endless runner game in Unity to keep the player challenged and not bored, was the main goal and purpose of this project.

In conclusion, the goal was:

- Implement dynamic difficulty adjustment
- Implement procedural content generation
- Make dynamic difficulty adjustment & procedural content generation work together.

## 1.2 Limitations

The task was not very clear at the start because I lacked knowledge of dynamic difficulty and L8 were unsure what exactly they wanted for their game. The Endless runner was recently started and was therefore in its early stages, so there were not so many game obstacles to choose from and several gameplay changes were made throughout the weeks. As the game is an endless runner, the difficulty variations to choose from is very limited because the gameplay of an endless runner is fairly uncomplicated.

## 1.3 Level Eight

Level Eight (L8) is an independent game development company located in Umeå, where they, during this project, employed 15 people [10]. It was founded in 2011 by a small core group of eight people who already had extensive experience with handheld gaming and during this project developing games based on the original IPs for the Apple iOS and Google Android platforms. Level Eight has developed 5 games over the past years and their most popular and famous game with over one million daily users is the Robbery Bob series [11]. Their YouTube channel is also quite popular with over 233,000 subscribers and ~41 million views in total [27]. All of the numbers above were accurate during the writing of this project. L8's logotype and Robbery Bob can be seen below in figure 1.1 and 1.2.



---

Figure 1.1 & 1.2: Level Eight's logotype, left figure [10] and Robbery Bob image, right figure [10]

## 1.4 Background

DDA can be seen as early as 1975, in the game Gun Fight by Midway Manufacturing Co [1]. The game would aid whichever player had just been hit by a shot, by placing an additional object on their side to make it easier for them to hide behind objects the next round. Another early game is Archon [24], from 1983. The computer opponent in Archon slowly adapts over time to help players defeat it.

Procedural level generation in video games existed as early as 1978 when Beneath Apple Manor [31] was released and Rogue [30] in 1980. They were one of the first to use procedural generation to construct dungeons for ASCII- or regular tile-based systems [2].

### 1.4.1 Endless Runner

“Endless running” or “infinite running” games are platform games in which the player character is continuously moving forward through a usually procedurally generated, theoretically endless game world [26], as shown in figure 1.3.



---

Figure 1.3: Screenshot of the endless runner

The endless runner is well-suited to the small set of controls, they are limited to making the character jump over obstacles, roll under obstacles and slide between different lanes. The main object of endless runners games is to get as far as possible before the character dies because of crashing into an object. As the character runs down the track, the player must collect coins, powerups and avoid obstacles through a combination of the controls mentioned above as the running speed slowly increases.

#### 1.4.2 Unity3D

Unity is a cross-platform game engine developed by Unity Technologies, which is primarily used to develop both two- and three-dimensional video games and simulations for computers, consoles and mobile devices [23]. The online documentation is good and detailed, thanks to the international use of the game engine.

#### 1.4.3 Atlassian - SourceTree

Atlassian Corporation [12] is an Australian enterprise software company that develops products for software developers, project managers, and content management. One of the products is SourceTree which is a powerful Git and Mercurial desktop client for developers on Mac or Windows. It simplifies how you interact with your Git repositories so you can focus on coding. Visualize and manage the repositories through SourceTree simple Git GUI.

#### 1.4.4 Flow

One of the most common approaches to the psychology of the optimal experience is the 'Flow' by M.Csikszentmihalyi [8] [14]. The 'Flow' [6] is the state of consciousness where an individual experiences the peak enjoyment of fulfillment while doing an activity, as described by psychologist Mihaly [8]. The goal of the Flow is to keep the player challenged, interested and not feeling anxious or bored by adjusting the game experience [6], as seen in figure 1.4. But it is not as simple as adding health back when the player is in a bad situation, it is a design problem that involves estimating when and how to interfere with the game.

*"The best moments in our lives are not the passive, receptive, relaxing times... The best moments usually occur if a person's body or mind is stretched to its limits in a voluntary effort to accomplish something difficult and worthwhile."* - Mihaly Csikszentmihalyi [8]

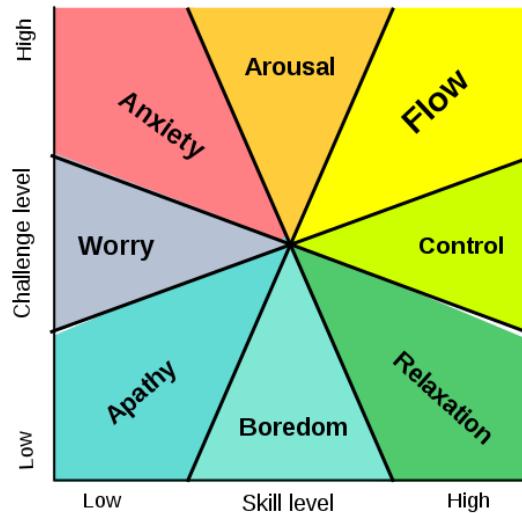


Figure 1.4: The mental state in terms of challenge level and skill level [6]

## 1.5 Related Work

Some examples of games that have taken advantage of DDA or PCG are explained below.

### 1.5.1 Borderlands

The very successful game series Borderlands [15] shows that procedural content generation can even be used in AAA productions to generate more diversity in the gameplay. Borderlands has a countless number of different weapons that are generated procedurally by pieces as shown in figure 1.5. By doing this they create an illusion of almost unlimited technical variety, making the game experience better.



Figure 1.5: Various weapon components that can be combined [28]

### 1.5.2 Crash Bandicoot

Dynamic difficulty adjustment [17] is a system introduced in Crash Bandicoot 2: Cortex Strikes Back [16] and then also implemented in many later Crash Bandicoot platforming games [17], which is designed to help players struggling with a certain level or section of a level. If a player dies repeatedly in a certain section of a level, Crash will be given a ‘powerup’ and will continue to do so on subsequent deaths until reaching the next checkpoint. Also, there are some crates in various levels that are programmed to become checkpoints after several deaths [17].

### 1.5.3 Infinite Mario Bros

Infinite Mario Bros [4] uses dynamic difficulty adjustment and procedural content generation. One of the inputs to the system is a set of parameters which specify probabilities for a specific event to occur, such as placing a gap. By modifying these probabilities, the system can create a large amount of levels with varying difficulty. So IMB scales up the difficulty by increasing the frequency of gaps, the average size of gaps, a variation of ground height, and the number of enemies.

### 1.5.4 Left 4 Dead

Left 4 Dead [18] has a system called ‘The Director’ that features a dynamic system for game dynamics, pacing, and difficulty. Instead of having set spawn points for all the zombies (enemies), the Director places zombies in varying positions and numbers based upon each player’s current location, skill, status, situation, creating a new experience for each play-through.

### 1.5.5 Mario Kart

A well-known example is a racing go-kart game called ‘Mario Kart’ [19] made by Nintendo [20]. Mario Kart uses a technique called ‘rubber-banding’ in a few different ways. One of them is adjusting the opponents’ speed depending on the player’s performance. If the player is doing well, all the opponents will speed up, if the player is not doing well, the opponents will slow down. Another adjustment is the amount and types of power-ups that are available to the player during a race. If the player is not doing well, the player will get more and better power-ups.

## 1.6 Method

The work process during this project consists of a few phases. The first phase was to research, learn and understand more about the main subject and create an overview over the process. This involved reading a lot of relevant blogs, forums, and papers, such as “Staying in the flow using procedural content generation and dynamic difficulty adjustment” by Parekh, R [3]. Dynamic difficulty adjustment is usually built for just one specific game so there is no universal ‘*this is how everyone does it*’ or a specific algorithm that everyone uses that can be applied to every game.

After the research phase, the focus was to get familiar with the endless runner and unity engine. The engine is fairly easy to acquire a general sense of how to use, but the game was in early development stage so changes were made continuously every day, so playing the game at least once a day was helpful, and take a brief look at the code.

The final phase was finding what different kind of variables I had to keep an eye on in order to calculate and implement the player performance which I could then use to adjust the difficulty, and also build chunks with varying difficulties.

## 1.7 Social, Ethical, and Environmental Considerations

### 1.7.1 Social considerations

One of the largest social impacts would be the same as with many other games. If the user gets addicted to the game it may have a negative impact on the person's personal life, such as sleep deprivation or disrupted food cycle.

### 1.7.2 Ethical considerations

The application saves no user information and this is at no risk of exposing any. No personal or sensitive information such as name, address etc. was collected. Dynamic Difficulty Adjustment will also make the game open to a broad player base because it adapts to every individual player.

### 1.7.3 Environmental considerations

Since the system is created for a game that is released on a digital platform it lessens the need to make physical copies and the game will require very low specifications so there will most likely be no need to purchase a new mobile or device.

## 2 Design and Implementation

The general design of the implementation was to create a dynamic difficulty adjustment system combined with a procedural content generator that would work in Unity3D. It had to be written in the programming language C# because the rest of the company is using C#.

### 2.1 Dynamic Difficulty Adjustment

Dynamic difficulty adjustment is a general term for methods that alter the gameplay to fit the player's performance. DDA is also known as Dynamic Game Balancing (DGB) or Dynamic Game Difficulty Balancing. More in-depth it is the process of automatically changing parameters, scenarios, and behaviors in a game in real-time based on the player's ability & skills. In order to avoid making the player bored (if the game is too easy) or frustrated (if it is too hard). The main goal of dynamic difficulty adjustment is to keep the user interested from the beginning to the end, providing a good level of challenge throughout the whole level. There is no universal algorithm for DDA because it is always designed depending on the type of game.

Deciding what game parameters to take into consideration for calculating players performance is a difficult task and needs considerable experimentation because the result may look very different depending on the parameter. Most games have different kind of parameters and there are only a few available in an endless runner game but some of them are the following:

- The number of coins presented.
- The number of coins collected.
- The number of powerups presented.
- The number of powerups collected.
- The number of successful dodges.
- Total distance traveled.
- Time alive.

But it is very hard to take all of the parameters into account and create something balanced out of it, so there should only be one primary unit of a parameter. Therefor, in this thesis, coins will be the main parameter, so the rest of the parameters are taken in terms of the main parameter. Some game elements that might be manipulated for dynamic difficulty in an endless runner are:

- The speed of the character.
- The frequency of obstacles.
- The frequency of powerups.
- The frequency of enemies (None in this game).
- Powerups duration

The higher speed the character has, the less reaction time the player gets, which makes it more difficult. Same with the frequency of obstacles, if there are more obstacles the likelihood of crashing into something will increase. Meanwhile increasing the chance of a powerup appearing will help and make the game less difficult for the player.

$$Performance = function(Skill, Difficulty) \quad (2.1)$$

Performance can be defined as a function of skill and difficulty. Above in equation 2.1, you can see the players performance (P) defined mathematically as a function of a given skill (S) and difficulty (D). Which can be used to identify the current difficulty zone.

To calculate the performance we need to add some penalty. Every 10 seconds a penalty is calculated, based on how many coins and powerups you picked up and missed. The penalty can be formulated as equation 2.2:

$$\begin{aligned} Penalty &= (20\% \text{ of total coins collected} * \text{number of times powerup lost}) \\ &\quad + (5\% \text{ of total coins collected} * \text{number of coins lost}) \end{aligned} \quad (2.2)$$

After the penalty, the performance of the player is directly calculated, given as equation 2.3:

$$Performance = ((Total \text{ coins collected} - Penalty) / Total \text{ coins presented}) * 100 \quad (2.3)$$

There is a penalty for every death depending on the time you stayed alive, the shorter you survived, less difficult. The longer you survived, more difficult. Also the multiplier increases as the more deaths you have, an example can be given as equation 2.4:

$$\varphi = (\text{Time alive} < \text{Lower threshold})$$

$$\Psi = (\text{Deaths} * 0.25f)$$

$$\Tau = -(\text{Deaths} * 0.1f)$$

$$(\varphi \rightarrow \Psi) \wedge (\neg\varphi \rightarrow \Tau) \quad (2.4)$$

To receive a more accurate difficulty we take the average of three performance calculations and if the average is between some thresholds we decrease or increase the difficulty, so the final calculation can be seen as equation 2.5.

$$\alpha = (\text{Average performance} > \text{Increase difficulty threshold})$$

$$\beta = \text{Increase the difficulty}$$

$$\gamma = \text{Decrease the difficulty}$$

$$(\alpha \rightarrow \beta) \wedge (\neg\alpha \rightarrow \gamma) \quad (2.5)$$

## 2.2 Procedural Content Generation

PCG is an approach to create content automatically with the use of algorithms instead of manual effort which contributes to effective resource usage and expandability. Having the game build the whole level by itself instead of placing it all manually saves a lot of time and work and you get more replayability because of the random factors.

The first main idea was to randomly spawn content all over the lanes based on Perlin Noise [21] and the difficulty provided by the dynamic difficulty calculations. The big issue with spawning content randomly all over the lanes, is that you need an enormous amount of different conditions to make sure the level is playable. This is because the contents are likely to spawn in such a way it blocks all possible paths. After some discussions and research by studying games like Subway Surfers [29], a new decision was made, building the levels with chunks. Chunks are hand-built sequences that can be seamlessly spawned after each other and because they are

built by hand they can never be impossible to complete (unless you build them impossible of course). The chunks will also appear as random because you will never know when that specific chunk is appearing. The chunks are built with a custom '*Chunk Editor*' where you start off with an empty grass field, seen in figure 2.7. You can then increase or decrease the length of the grass field and also place the obstacles, coins or powerups at a desirable position, as figure 2.6.

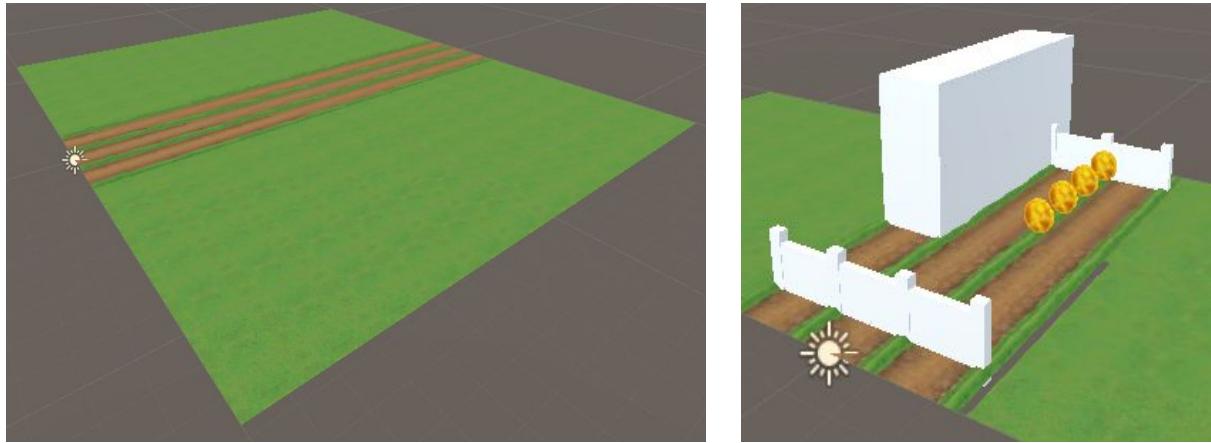


Figure 2.6 & 2.7: Grass field with obstacles, figure left and empty grass field, figure right.

The chunks are exported and imported in the form of XML files as in figure 2.8.

```
<?xml version="1.0" encoding="Windows-1252"?>
<ChunkCollection xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
<Objs>
  <Obj type="400" x="1.5" z="9" />
  <Obj type="1100" z="5" />
  <Obj type="300" x="-1.5" z="6" />
</Objs>
<Cls>
  <Cl type="200" x="1.5" z="4" />
</Cls>
<Length>3</Length>
<Difficulty>1</Difficulty>
</ChunkCollection>
```

 chunkEasy.xml

Figure 2.8: XML Chunk in notepad++

The parameters in the XML file has different meanings:

- **Obj type** - Defines what enum type of obstacle.
- **Cl type** - Defines what enum type of pickup.
- **Length** - The length of the chunk.
- **Difficulty** - The difficulty of the chunk.
- **x, y, z** - position in x, y, z-axis of the obstacle / pickable.

## 2.3 Implementing the System

The implementation of this project was made in C# with the engine Unity3D, inside a branch based on the development-branch, so if something would break then no one else would become affected and whenever there are updates on the development-branch it's easy to pull the new commits.

First, a variable manager is made to keep track of all the different variables. The ones that are kept track of are the following:

- **int\_score** - Based on distance traveled.
- **int\_coins** - Coins picked up by the player
  - A script that checks if the player has run past (missed) or collected the coin, is placed on the coin-prefab.
- **int\_coinsMissed** - Coins missed by the player.
- **int\_coinsTotal** - Total amount of coins.
  - \_coinsTotal = \_coins + \_coinsMissed.
- **int\_powerup** - Powerups picked up by the player.
  - A script that checks if the player has run past (missed) or collected the powerup, is placed on the powerup-prefab.

- ***int\_powerupMissed*** - Powerups missed by the player.
- ***int\_powerupTotal*** - Total amount of powerups.
  - $_powerupTotal = _powerup + _powerupMissed$ .
- ***int\_deaths*** - Total amount of player deaths.
  - An event listener to check if the player has died.
- ***int\_timeAlive*** - The time the player has been alive in the current round.
  - If the player is alive,  $+Time.deltaTime$
- ***int\_timeAliveAverage*** - The average time stayed alive in a round.
  - Every single *\_timeAlive* is put in a list and divided by the size of the list.

All of the above variables are declared as private because the ‘\_’-prefix indicates it as private, but they have public getters & setters to easily set or return a value outside of the variable manager. The manager also contains two reset-functions for the variables. ‘*resetStats*’ that sets every single variable to zero and ‘*resetStatsForPerformance*’ that sets variables specific to the performance calculation to zero.

Next up is building the dynamic difficulty adjustment structure. What we need to calculate is the penalty, the performance, and the difficulty.

To calculate the difficulty we need the performance, and to calculate the performance we need to calculate the penalty. So first we start off by calculating the penalty, and we want to do this every 10 seconds. The variables we will need for this is:

- The number of coins collected
- The number of coins missed
- The number of total coins presented
- The number of missed powerups

Pseudo-code for the penalty implementation in C# can be seen in figure 2.9.

```
InvokeRepeating("calculateThePenalty", 10, 10); // Run every 10  
seconds  
  
private int calculateThePenalty() {  
    float powerupPenalty = (20% of coins collected) * powerups missed;  
    float coinPenalty = (5% of coins collected) * coins missed;  
    int penalty = Mathf.RoundToInt (powerupPenalty + coinPenalty); // Round to nearest int  
    return penalty;  
}
```

---

Figure 2.9: Pseudo-code for penalty implementation

After we have calculated the penalty we continue by calculating the performance which can be implemented as figure 2.10.

```
private int calculateThePerformance (int penalty) {  
    float perf = coins collected - penalty;  
    perf = perf / total coins presented;  
    perf = perf * 100.0f; // Change range to 0-100  
    int performance = Mathf.RoundToInt (perf) // Round to nearest int  
  
    resetStatsForPerformance (); // Reset stats for next calculation  
    return performance;  
}
```

---

Figure 2.10: Pseudo-code for performance implementation

The performance explains how good the player performed on a range 0 to 100 based on the coins and powerups. Next up, we want to store the performance inside a list, and once the list reaches a size of three, run the '*chooseDifficulty*' function as in figure 2.11.

```

private void chooseDifficulty() {
    for ( int i = 0; i < _performanceList.Count; i++ ) {
        averagePerformance += _performanceList[i];
    }
    averagePerformance = averagePerformance / _performanceList.Count;
    _performanceList.Clear();

    if ( averagePerformance > increase difficulty threshold ) {
        Increase the difficulty
    }
    else {
        Decrease the difficulty
    }
}

```

---

Figure 2.11: Pseudo-code for difficulty implementation

Then, with the difficulty provided by '*chooseDifficulty*' we either increase or decrease the level difficulty based on thresholds.

Last off is the content generation. As randomly spawning content all over the lanes is very hard to accomplish we chose to go as previously discussed with building chunks by using an in-house '*Chunk editor*'. With the chunk editor, we can increase or decrease the length of the chunks with a single button, place obstacles, coins or powerups wherever we decide. Secondly, set a difficulty on the chunks based on how they are built. A lot of obstacles usually means it is harder and fewer obstacles usually means it is easier. So now we can spawn chunks in real-time based on how the player is currently performing.

# 3 Results

The goal of the project was to implement dynamic difficulty adjustment with procedural content generation in Level Eight's endless runner. A player would play the game for a while, the dynamic difficulty adjustment system calculated the performance of the player and then with the help of the dynamic content generator spawn levels according to the player performance. At the end of the implementation stage, this was the result that we ended up with. The result differed some from the main idea that was planned in the beginning.

## 3.1 End Results

In the final version, the dynamic difficulty adjustment system could calculate the player performance based on coins and powerups collected with coins as the main parameter. Then adjust the difficulty of the upcoming chunks by using the performance provided.

## 4 Discussion

The task of creating a working dynamic difficulty adjustment system with procedural content generation can result in many different ways depending on the game. The goal that was set at the beginning of the project was achieved and worked as expected. Though the first idea was to create a procedural content generator but as some time passed and work had been done we realized the outcome of randomly spawning content all over the lanes wouldn't be appealing to the player, and probably very hard to make playable. So the "randomly spawning content all over the lanes"-project was discarded and a new idea was brought to life. The idea was building small chunks by hand and spawn them seamlessly based on their difficulty which in the end is a more solid solution.

All the chunk difficulties are set by a personal opinion which can have an impact on the gameplay because what the creator sets as a 5 (medium) might feel like a 10 (hard) for someone else. So the result of doing this can be that the player dies early even though that wasn't planned because the chunk is too hard.

### 4.1 Future work

There is of course always room for further improvements and additions. Some future work for this implementation could be:

- Increase difficulty length.
  - Increase the length of the difficulty from 1-10 to a wider range such as 1-100, so we can receive a more accurate and deep difficulty because the 1-10 range only contains 10 difficulty options meanwhile 1-100 has 100 options. Or even change the difficulty to a real number instead of an integer, in order to get an infinite amount of alternatives.
- Increase complexity.
  - The algorithms are fairly simple and could be made more complex and have more contributing factors. This could be both a bad and a good idea depending on the amount of complexity.

- Amount of chunks.
  - The more chunks that are available the more we have to choose from, which will increase the variation.
- Chunk difficulty based on user-input
  - Currently, all the chunk difficulties are set by a single opinion, which isn't accurate at all. A more solid idea would be having a couple of people, play some specific chunks at 50% speed, and afterward they rate the difficulty. Then take an average range of the inputs.

## 5 Conclusion

In this thesis a dynamic difficulty adjustment system was presented by considering several conditions of the player. This was done with the help of the game environment by using procedural content generation to ensure the game is balanced yet still welcoming to beginners and challenging for advanced players respectively. The player's stats were used to determine and affect the difficulty of the game, some of the collected stats were total coins collected, coins missed, powerups collected, powerups missed.

## 6 References

- [1] Wikipedia. (2018). Dynamic Difficulty Balancing. Available at:  
[https://en.wikipedia.org/wiki/Dynamic\\_game\\_difficulty\\_balancing](https://en.wikipedia.org/wiki/Dynamic_game_difficulty_balancing) [Accessed 2018-04-06]
- [2] Lee, J. (2014). How procedural generation took over the gaming industry. Available at:  
<https://www.makeuseof.com/tag/procedural-generation-took-gaming-industry/> [Accessed 2018-04-06]
- [3] Parekh, R. (2017). Staying in the flow using procedural content generation and dynamic difficulty adjustment. Available at:  
<https://web.wpi.edu/Pubs/ETD/Available/etd-042717-113745/unrestricted/raparekh.pdf>  
[Accessed 2018-04-06]
- [4] Lewis, C. (2010). Infinite Mario Bros. Available at:  
<https://github.com/cflewis/Infinite-Mario-Bros> [Accessed 2018-04-10]
- [5] Patel, A. (2013). Noise Functions and Map Generation. Available at:  
<https://www.redblobgames.com/articles/noise/introduction.html> [Accessed 2018-04-09]
- [6] Csíkszentmihályi, M. (1990). Flow: The Psychology of Optimal Experience. Harper & Row.. ISBN: 978-0-06-016253-5.
- [7] José, R & Filipe T. (2014). Procedural Level Balancing in Runner Games. Available at:  
<http://www.sbgames.org/sbgames2014/files/papers/computing/full/303-computingfullpages.pdf> [Accessed 2018-04-16]
- [8] Oppland, M. (2016). Mihaly Csikszentmihalyi: All about flow & positive psychology. Available at: <https://positivepsychologyprogram.com/mihaly-csikszentmihalyi-father-of-flow/> [Accessed 2018-04-16]
- [9] Saltsman, A. (2009). Game Changers: Dynamic Difficulty, Available at:  
[https://www.gamasutra.com/blogs/AdamSaltsman/20090507/83913/Game\\_Changers\\_Dynamic\\_Difficulty.php](https://www.gamasutra.com/blogs/AdamSaltsman/20090507/83913/Game_Changers_Dynamic_Difficulty.php) [Accessed 2018-04-16]
- [10] Level Eight. (2018). Level Eights homepage. Available at: <http://www.leveleight.se/>  
[Accessed 2018-04-02]
- [11] Level Eight. (2018). Robbery Bob: Man of Steal. Available at:  
<http://www.leveleight.se/products/robbery-bob/> [Accessed 2018-04-02]

[12] Atlassian Corporation. (2018). Atlassian homepage. Available at:  
<https://www.atlassian.com/> [Accessed 2018-04-17]

[13] Atlassian Corporation. (2018). Atlassian SourceTree. Available at:  
<https://www.sourcetreeapp.com/> [Accessed 2018-04-17]

[14] Claremont Graduate University. (2018). Mihaly Csikszentmihalyi. Available at:  
<https://www.cgu.edu/people/mihaly-csikszentmihalyi/> [Accessed 2018-06-02]

[15] Gearbox Software. (2018). Borderlands the game. Available at:  
<https://borderlandsthegame.com/> [Accessed 2018-04-17]

[16] Bandipedia. (2018). Crash Bandicoot 2: Cortex Strikes Back. Available at:  
[http://crashbandicoot.wikia.com/wiki/Crash\\_Bandicoot\\_2:\\_Cortex\\_Strikes\\_Back](http://crashbandicoot.wikia.com/wiki/Crash_Bandicoot_2:_Cortex_Strikes_Back) [Accessed 2018-04-17]

[17] Bandipedia. (2018). Crash Bandicoot Dynamic Difficulty. Available at:  
[http://crashbandicoot.wikia.com/wiki/Dynamic\\_Difficulty\\_Adjustment](http://crashbandicoot.wikia.com/wiki/Dynamic_Difficulty_Adjustment) [Accessed 2018-04-17]

[18] Thompson, T. (2015). In the directors chair: Left 4 Dead. Available at:  
<https://aiandgames.com/in-the-directors-chair-left-4-dead/> [Accessed 2018-06-02]

[19] Nintendo. (2018). Nintendo - Mario Kart. Available at: <https://mariokart8.nintendo.com/>  
[Accessed 2018-06-02]

[20] Nintendo. (2018). Nintendo homepage. Available at: <https://www.nintendo.com/>  
[Accessed 2018-04-17]

[21] Zucker, M. (2001). What is Perlin Noise? Available at:  
<https://mzucker.github.io/html/perlin-noise-math-faq.html#whatsnoise> [Accessed 2018-06-02]

[22] Slashdot. (2010). Infinite Mario With Dynamic Difficulty Adjustment. Available at:  
<https://games.slashdot.org/story/10/09/08/055245/infinite-mario-with-dynamic-difficulty-adjustment> [Accessed 2018-04-17]

[23] Unity. (2018). Unity3D home page. Available at: <https://unity3d.com/> [Accessed 2018-04-19]

[24] Edwards, Benj. (2005). The Secrets of Archon. available at:  
<http://www.vintagecomputing.com/index.php/archives/44> [Accessed 2018-04-26]

[25] Slashdot. (2009). Should Computer Games Adapt To the Way You Play. Available at:  
<https://games.slashdot.org/story/09/10/13/078247/Should-Computer-Games-Adapt-To-the-Way-You-Play> [Accessed 2018-05-02]

- [26] Chong, B. (2015). Endless Runner Games: How to think and design (plus some history). Available at:  
[https://www.gamasutra.com/blogs/BenChong/20150112/233958/Endless Runner Games Ho w to think and design plus some history.php](https://www.gamasutra.com/blogs/BenChong/20150112/233958/Endless%20Runner%20Games%20How%20to%20think%20and%20design%20plus%20some%20history.php) [Accessed 2018-06-02]
- [27] Youtube. (2018). Level Eights Youtube Channel. Available at:  
<https://www.youtube.com/user/LevelEightVideos/about> [Accessed 2018-05-03]
- [28] Borderlands Wiki. (2015). Weapons. Available at:  
[http://borderlands.wikia.com/wiki/Weapons#cite\\_note-chimeric-rifle-1](http://borderlands.wikia.com/wiki/Weapons#cite_note-chimeric-rifle-1) [Accessed 2018-05-15]
- [29] Kiloo. (2017). Subway Surfers. Available at: <https://subwaysurfers.com/> [Accessed 2018-05-20]
- [30] Moby Games. (2018). Rogue. Available at: <https://www.mobygames.com/game/rogue> [Accessed 2018-05-31]
- [31] Worth, D. (2018). Beneath Apple Manor. Available at:  
<http://www.mobygames.com/game/beneath-apple-manor> [Accessed 2018-05-31]

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/327920069>

# Dynamic difficulty adjustment through parameter manipulation for Space Shooter game

Conference Paper · September 2015

---

CITATIONS

3

READS

240

3 authors:



Renê Pereira de Gusmão  
Federal University of Sergipe

6 PUBLICATIONS 15 CITATIONS

[SEE PROFILE](#)



Kennet Calixto

3 PUBLICATIONS 10 CITATIONS

[SEE PROFILE](#)



Caetano Segundo

2 PUBLICATIONS 10 CITATIONS

[SEE PROFILE](#)

Some of the authors of this publication are also working on these related projects:



Clustering of multi-view data [View project](#)

# Dynamic difficulty adjustment through parameter manipulation for Space Shooter game

Caetano Vieira Neto Segundo<sup>1\*</sup>

Kennet Emerson Avelino Calixto<sup>1</sup>

Renê Pereira de Gusmão<sup>2</sup>

<sup>1</sup> Faculdade Paraíso - CE  
Departamento de Sistemas de Informação  
Brazil  
<sup>2</sup> Universidade Federal de Sergipe  
Departamento de Computação  
Brazil

## ABSTRACT

An important criteria to be considered in the development of games is the game experience. It is directly linked to the difficulty of the game, which can be in the static or dynamic way. Players with different abilities in the same game may not fit in traditional difficulties models. The dynamic difficulty adjustment (DDA) provides a better gameplay, as the challenges in the game fit the player's abilities. Thus, this work proposes the development of the parameter manipulation technique for dynamically adjusting difficulty, aiming to improve the gaming experience. It is necessary to emphasize that the proposed approach uses probabilistic calculations that will be used in the challenge function. A questionnaire was applied to a sample of students in order to determine whether there were statistically significant differences in the perception of game play, difficulty of the game and desire to play several times with and without the use of the technique. The results showed that the dynamic version was better evaluated regarding game play and appropriate difficulty when compared to easy and hard versions.

**Keywords:** Artificial Intelligence, Dynamic Difficulty Adjustment, Digital games, Space Shooter, Parameter Manipulation.

## 1 INTRODUCTION

In the development of a game, a very important factor that should be assessed is the difficulty level and the types of challenges created for the player. In general, players choose the difficulty level they want to play at. Most of them see it as an obstacle the fact estimate their own abilities, and then end up playing a very easy or difficult game. Some of these problems are caused by the following: little variation in levels, gaps between one level and another and the lack of engaging challenges for the player as explained by Arulraj [9].

The different skills between players, of the same game, make the traditional model difficult to balance because it does not offer a level everyone feels comfortable with. This may be a problem because it can discourage many players. The users deception can cause them to have a bad experience and making it unlikely for them to play a particular game.

In this sense, balancing is a key factor in ensuring a pleasant gaming experience. This balancing is performed by the game designer, who performs an exhaustive amount of testing to find the features that represent the appropriate levels of the game as explained in Andrade, Ramalho and Santana [1].

In Hunicke [6], it's described that the dynamic balancing game consists of a mechanism that assesses the users performance in order to observe the challenges posed by the game. Then it automatically

makes adjustments to the game in order to match the capacity of the player. On the other hand the static balancing is a traditional way of balancing, where the game has preset levels for all users and each of them chooses to start the match.

It is expected that the dynamic balancing provides a better game experience. The implementation of a DDA technique in the games provides consistent challenges to the players skills. Thus, it creates a fair scenario for the players. In this work the development of a dynamic balancing technique for the game Space Shooter will be discussed. The technique applied will be the manipulation of parameters. Where the challenge function will be defined through a probabilistic method that will be able to determine the situation of the player in the scenario.

The remainder of this work is structured as follows. Section 2 presents a review of some related works concerning dynamic difficulty adjustment applied to games. Section 3 presents an overview about the Space Shooter game and related concepts. Section 4 introduces the DDA technique applied to the Space Shooter. The results of the statistical analysis are shown in Section 5. The conclusion and future works are discussed in section 6.

## 2 RELATED WORK

In this section we discuss some related works. There are several works regarding dynamic difficult adjustment in digital games. Some of them are discussed below.

In Araujo and Feijo [2], the authors use the dynamic adaptively to evaluate casual and hardcore players, related to concepts of flow theory and the model core elements of the game experience. The authors also present an adaptive model for shooting games based on player modeling and online learning. The authors developed an implementation of a framework in Charles and Black [3] for player modelling and dynamic adaptivity. Two versions of the game were tested: an adaptive version using the implementation of the framework and a non-adaptive version. The results supported the idea that hardcore players had a better assimilation of the gaming experience with the adaptive version, as expected.

In Hunicke and Chapman [7], the authors explore and design requirements for a DDA system. They presented a probabilistic method for representing and reasoning about the uncertainty in games, also described the implementation of the proposed techniques and discussed how it can be applied to improve interactive experiences. In Hunicke [6], the author evaluated basic design requirements for effective DDA and presented an interactive DDA system. The study indicated that adjustment algorithms can improve performance, but retaining the players sense of agency and accomplishment.

The authors in Hawkins, Nesbitt and Brown [5] described a modeling technique known as particle filtering which is used to model several levels of the players ability while it can also consider the players risk profile since the performance in some games also de-

\*e-mail: caetanov120@gmail.com

pends on the players desire to take risk. The authors demonstrated the technique by creating a game challenge where the player can risk more, take less time and fail more or the player can risk less, take more time to evaluate and fail less.

In Sha *et al.* [11], the authors contributed to proposing DDA as an approach to create an appropriate challenge level game opponent, in which the game used was Dead-End. Two kinds of DDA were proposed. The first is DDA by time-constrained-CI which is based on Monte Carlo for tree search. The second is DDA by knowledge-based-time-constrained-CI which is based on artificial neural networks. The former more appropriate for standalone PC game and the latter more appropriate for multi-player online games.

### 3 DISCUSSION

In this section we present a discussion about Spacer Shooter game and some related concepts about game design and DDA.

#### 3.1 Space Shooter

The chosen game for the implementation of the technique was the Space Shooter, an action game made with Unity [12]. The choice was based on the few number of commands that the game requires to be played, which makes it faster learning of the user. This game can be found in the unity repository and all of its assets are available free of charge.

In Space Shooter game, the player has a ship cover of ammo and life (the player starts with 100 hit points and 50 bullets in your inventory), and he faced his enemies and obstacles, in order to collect points. Throughout the match packages will be offered, they can increase ammo or life when in contact with the player's ship. The punctuation of the player is added only when it killing your enemy with your ammo, if the ship collides with his enemies, the ship loses life and the points remain.

For the development of the game, we used the game engine Unity. Unity provides ease and speed for the creation of 2D and 3D games. And its multi-platform resource includes platforms such as iOS, Android, Windows, Linux, SmartTVs. Moreover, it offered the developer learning resources, community and documentation in Unity [12].

#### 3.2 Dynamic balancing

In literature, it proposes different techniques for DDA games. There are proposals that handle global mode game and others that work directly with the characters of the non-player characters game (NPC's), but that each of them can act correctly it is necessary to know the user level. To determine this level of difficulty, approaches found using a challenging call function. It aims to determine the difficulty experienced by the user at any given time of the game. According to the result of the function, the policy will be applied, which are aimed at setting the challenge to the player. For the application of any of a technical study of the problem is critical.

##### 3.2.1 Dynamic Scripts

Traditionally the NPC's has its behavior applied by the intelligent agents. The agent has several sensors like perceiving the environment and through its actuators, it can perform appropriate actions. The decision of the choice to be made is defined by the agent function. To understand the function, we created a table that describes the possible agent entries and each entry is recorded in Russell and Norvig [10].

##### 3.2.2 Genetic Algorithms

Some studies propose the use of this technique for balancing. Melanie [8] proposes the use of learning based on genetic algorithms through crossover and mutation are created new individuals.

In this context it will not be the best selected, but those who improve to adapt to the user level. This ability is given by the heuristic function.

#### 3.2.3 Parameter manipulation

This is the approach that was used in the present paper. It has a global control of the variables, i.e., the whole game can be controlled by the algorithm. In this model if we apply the algorithm correctly, you can adjust any parameter of the play structure at the time that the player is active as shown by Hunicke [6].

This technique uses an economic vision market. This economic model of supply and demand parameters goal is to keep everything balanced as best as possible.

In the DDA, the supply parameter is seen as the items that the game offers to the user (health, weapons) and also the parameters of the characters (life, strength and endurance). Demand is set by manipulating the parameters of enemies (force, weapons, behavior). The implementations of game policies should cover enemies with different levels of difficulty. Thus the algorithm will have more level options to be dynamically chosen in Hunicke and Chapman [7].

In Space Shooter the offer of parameters analyzed are: hit points and ammunition; the demand parameters are: the enemy's behavior and strength. The balance between the curves ensure that the level is appropriate difficulty felt by the player. Figure 1 shows the flow channel that the player must meet throughout the game, avoiding the undesirable states that are too difficult or too easy in Hunicke and Chapman [7].

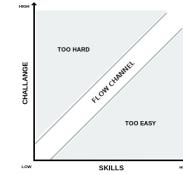


Figure 1: Difficult flow of the player, adapted from Hunicke and Chapman [7]

### 4 PROPOSED METHODOLOGY

In this section we introduce the methodology used for the dynamic difficult adjustment for the Space Shooter.

#### 4.1 Heuristic function

It is necessary to determine the user level in the game, we assume that the level varies according to the damage suffered in a given time ( $t$ ) in the game. We performed a probabilistic calculation with the damage of the player, and we have to answer an estimated death at a future time. In addition to the estimated death is associated with the amount of life the player to determine if the algorithm should act in some way.

The sum of the damages ( $d$ ) at a time is given by the function (1) where  $i$  represents the damage's value.

$$\sum_{i=1}^n d(i) \quad (1)$$

The set of times allows us to calculate the probability through the normal distribution function or Gaussian distribution, given by (2) function.

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-m}{\sigma})^2} \quad (2)$$

Although the Gaussian distribution manages the entire curve, it is necessary to calculate the mean and standard deviation, in which the mean is obtained by the equation (3). The calculation is done using a C# library that implements several methods. Meta Numerics supports mathematical and statistical calculations to the .NET platform, such as the C# proposed by CodePlex [4].

$$\mu = \frac{t_1 + t_2 + \dots + t_n}{n} = \frac{1}{n} \sum_{i=1}^n t(i) \quad (3)$$

## 4.2 Control zones

The area is the mechanism used to classify the level of the player at a given time instant. In this approach we used two areas: comfort and discomfort. Each has distinct characteristics and from them you can apply control policies.

In Figure 1, the comfort zone would be the part below the flow (very easy), so it would not be requiring any effort for the player, which may cause a lack of interest in the game. The top of the flow would be the discomfort zone (very difficult), it would be presenting challenges that exceed the ability of the player. The flow channel must be the site that the player will remain and the challenges are consistent with the ability of the same, but there is also unpredictability, since the player can change position within that stream without the algorithm reaction. Unpredictability becomes important for the game does not present repetitive challenges and end up discouraging the player.

Define the zone that the player belongs is essential for policies to be applied properly, however the values of each parameter should be studied and defined by a game design. In this work we do not discuss about the game design. In Space Shooter there are two areas according to the likelihood of damage and the percentage of the player's life. For the junction of these two variables are extremely important to know the level of difficulty that the player is feeling.

### 4.2.1 Comfort zone

When the player is in the comfort zone, it was decided that the likelihood of damage is less than or equal to 70% and its life percentage is greater than or equal to 60%. In the comfort zone, the player will be in a hostile area with more enemies, more damage and less support (life and bullet) of the game.

### 4.2.2 Discomfort Zone

This area defined the players who do not have many skills in the game, so that the user is characterized in that area, it was determined that the percentage living below 50% and the damage estimate over 55%. The player enters this area when he feels difficulty from the challenges posed by the game. In this area the player will have more support (life and bullet) of game, enemies in smaller quantities and with less damage.

### 4.2.3 Flow Zone or Unpredictability

In this case we have an ideal scenario, i.e., the player is between the two previously mentioned areas. When you are in that area, the algorithm will not take any measure related to user assistance.

In this scenario, the game will provide challenges according to the player's abilities. Thus, the algorithm will not provide any help according to the player's health and ammo demand, but will continue running the assessments to interfere in the future if necessary.

## 4.3 Policies

Policies are interventions that manipulate the game parameters in order to move the user through the channel, as shown in Figure 1. After the classification of the player in the region, a policy will be applied, and may increase or decrease the challenges of the game, or simply keep as being.

Adjustments can be in or out of the stage. The actions taken in, are handled at the time of the confrontation, and if necessary the parameters are changed (enemy's strength, life, ammo). Changes offstage, are performed when the enemies are being generated and may modify the order, the amount and the damage.

The actions taken by the game will only be noticed by the user in the next wave, ie, adjustments are made out of the game scenario. It is believed that this will ensure a good unpredictability, because although in certain the player to be helped, it is not desirable that the algorithm is making changes in the scenario at all times.

### 4.3.1 Settings

Adjustments are made through two functions. The first one is the assessment, which is performed by a subroutine of the game. In this function, the algorithm starts probabilistic calculation player death after 7 times of the game each time and set to 5 seconds. The function calculates the mean and standard deviation. Finally it is returned probability.

The second function is the challenge function that every 5 seconds performs a check of the values provided by the assessment and performs the necessary actions to adjust the difficulty regarding the characteristics of the areas in which the player is placed. The assessment and the adjustment is based on the probability of damage and life of the player. The adjustment can put the player in three different areas.

## 4.4 Example



Figure 2: Playing Space Shooter

In a practical example for three zones, as shown in the Figure 2, one can imagine a situation where the player has life in 40% and the probability calculated damage is approximately 70%, which makes higher the chances player die. At this point, the algorithm evaluation function feeds the challenge function, which will analyze the likelihood of damage and the players life consistent with the characteristics of the discomfort zone, and to help the player, the algorithm will provide a life box every 5 seconds and reduce the enemies' damage and amount. On the other hand, in a situation of the comfort zone, the algorithm would provide a life box every 20 seconds, increase the enemies damage and gradually the amount.

## 5 RESULTS

In this section we present the results found by the statistical analyses made from the results of questionnaires applied to the players.

### 5.1 Descriptive statistics

To evaluate the perception of individuals with regard the appropriate difficulty, the playability of the game and the desire to play the game several times, a questionnaire with questions assessing these factors was applied to 30 people. For this, people played the easy version of the game and answered the questions facing cited variables, doing the same for dynamic and difficult version of the game. The answer scale used to the questions ranged from 1 to 5, where 1 meant totally disagree with the statement and 5 meant totally agree with the statement.

Regarding the evaluation of the best version after playing the three versions of the game, the participants mostly (18 people, representing 60% of the sample) classified the dynamic version as the

best version of the game. On the other hand, 9 people evaluated the hard version as the best and 3 people evaluated the easy version as the best.

Regarding the score of the participants, half of them performed better in the easy version of the game, representing 50% of the sample. In the dynamic version, 15 of them (50% of the sample) also had better scores, which supports the assessment of participants who rated above the dynamic version as being the best. Thus, when people played the dynamic version they were as good as when they played the easy version of the game. For the hard version no one got better score, as expected.

## 5.2 Inferential statistics

In order to compare the easy and the dynamic versions of the game with regard to game play, appropriate difficulty and will play the game several times, some T test for paired samples were made in order that the same people participated the three experimental conditions (easy game, dynamic game, difficult game).

Table 1 presents the results of the comparisons between the easy and dynamic version of the game, in which GP means Game play, AD means appropriate difficult and PST means play several times.

Table 1: Easy version vs. dynamic version

|     | Easy |      | Dynamic |      | t(df)     | p     | 95% CI      |
|-----|------|------|---------|------|-----------|-------|-------------|
|     | M    | SD   | M       | SD   |           |       |             |
| GP  | 4,03 | 0,76 | 4,27    | 0,69 | -2,25(29) | 0,032 | -0,45;-0,02 |
| AD  | 2,40 | 1,25 | 4,30    | 0,60 | -8,78(29) | 0,001 | -2,34;-1,46 |
| PST | 2,63 | 1,35 | 4,03    | 0,76 | -5,66(29) | 0,001 | -1,91;-0,89 |

Regarding the game play, the test results showed a statistically significant differences regarding game play easy and dynamic version of the game ( $t(29) = -2.25, p < 0.05$ ). In this case, the participants evaluated the dynamic version as better ( $M = 4.27, SD = 0.69$ ) compared to the easy version of the game ( $M = 4.03, SD = 0.76$ ). Thus, it can be said that the dynamic version was evaluated as the one best to play, comparing the easy version of the game.

The results of t-test showed that there are statistically significant differences in the assessment of appropriate game difficulty ( $t(29) = -8.78, p < 0.001$ ). The dynamic version has more difficulty ( $M = 4.30, SD = 0.60$ ) compared adequate easy version of the game ( $M = 2.40, SD = 1.25$ ). Therefore, the participants believe that the dynamic version has a better difficulty, or better play.

Regarding the willingness to play the game repeatedly, the t test showed that there is also significant differences in the two versions( $t(29) = -5.66, p < 0.001$ ). The participants felt more desire to play the dynamic version ( $M = 4.03, SD = 0.76$ ) instead of the easy version ( $M = 2.63, SD = 1.25$ ).

Table 2: Hard version vs. dynamic version

|     | Hard |      | Dynamic |      | t(df)     | p     | 95% CI       |
|-----|------|------|---------|------|-----------|-------|--------------|
|     | M    | SD   | M       | SD   |           |       |              |
| GP  | 3,97 | 0,96 | 4,27    | 0,69 | -2,07(29) | 0,048 | -0,60;-0,003 |
| AD  | 2,60 | 1,45 | 4,30    | 0,60 | -5,53(29) | 0,001 | -2,33;-1,07  |
| PST | 3,70 | 1,34 | 4,03    | 0,76 | -1,33(29) | 0,194 | -0,85;-0,18  |

Table 2 presents the results comparing the hard version with the dynamic version. For the gameplay, the results of t test revealed a statistically significant difference when comparing the version difficult with the dynamic version ( $t(29) = -2.07, p < 0.05$ ). The participants assessed the dynamic version ( $M = 4.27, SD = 0.69$ ) as the best gameplay compared to hard version ( $M = 3.97, SD = 0.96$ ) of the game.

Regarding the appropriate difficult for these versions, there was also statistically significant difference ( $t(29) = -5.53, p < 0.001$ ).

Participants reported that the dynamic version ( $M = 4.30, SD = 0.60$ ) has more appropriate difficulty compared to hard version of the game ( $M = 2.60, SD = 1.45$ ).

For the desire to play the game several times, the difficult version ( $M = 3.70, SD = 1.34$ ) did not differ from the dynamic version of the game ( $M = 4.03, SD = 0.76$ ), ( $t(29) = -5.66, p < 0.001$ ). The participants would like to repeatedly play the two, with no statistically significant differences in this desire, even if the dynamic version has been evaluated with a higher average.

## 6 CONCLUSION

This paper proposed the use of parameter manipulation technique for dynamic difficulty adjustment applied to the game space shooter. A sample of people tested the game in three difficulties, two of them static (easy and hard) and the third dynamic. After playing in three versions, one questionnaire was used to assess the perception of people as the game play, difficulty and will play several times the game for each version.

A statistical analysis was performed to determine whether there were statistically significant differences in the responses of the questionnaires. The results of this analysis showed that the version with dynamic balancing achieved the best results when compared to the other versions regarding the game play and will play several times. As the score, half the people in the sample had better scores in the easy version and the other half had the best score in the dynamic version.

The game is in an initial version and can be improved in future. The improvements can serve for academic purposes for maximize the understanding, demonstrate and apply the method of DDA using the parameter manipulation. As future works, we suggest resources such as new phases for the game, new enemies and improve the interface, offering this ways a better game experience. We also suggest identify the viability of other techniques of dynamic balancing.

## REFERENCES

- [1] G. Andrade, G. Ramalho, H. Santana, and V. Corruble. Extending reinforcement learning to provide dynamic game balancing. In *Proceedings of the 2005 IJCAI Workshop on Reasoning, Representation, and Learning in Computer Games*, pages 7–12, 2005.
- [2] B. B. P. L. Araujo and B. Feijo. Evaluating dynamic difficulty adaptivity in shootem up games. In *SBC Proceedings of SBGames 2013*, pages 229–238, 2013.
- [3] D. Charles and M. Black. Dynamic player modeling: A framework for player-centered digital games. In *International Conference on Computer Games: Artificial Intelligence, Design and Education*, page 2935, 2004.
- [4] CodePlex. The meta.numerics math and statistics library.
- [5] G. Hawkins, K. Nesbitt, and S. Brown. Dynamic difficulty balancing for cautious players and risk takers. *International Journal of Computer Games Technology*, 2012, 2012.
- [6] R. Hunicke. The case for dynamic difficulty adjustment in games. In *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology*, pages 429–433, 2005.
- [7] R. Hunicke and V. Chapman. Evaluating dynamic difficulty adaptivity in shootem up games. In *Challenges in Game Artificial Intelligence AAAI Workshop*, pages 91–96, 2004.
- [8] Melanie Mitchell. An introduction to genetic algorithms. pages 4–23, 1999.
- [9] A. Joy James Prabhu. Adaptive agent generation using machine learning for dynamic difficulty adjustment. *IEEE*, pages 746 – 751, 2010.
- [10] J. Stuart Russell and P. Norvig. Artificial intelligence: A modern approach. pages 31–33, 2013.
- [11] L. Sha, S. He, J. Wang, J. Yang, Y. Gao, Y. Zhang, and X. Yu. Creating appropriate challenge level game opponent by the use of dynamic difficulty adjustment. In *Sixth International Conference on Natural Computation*, pages 3897–3901, 2010.
- [12] Unity Technologies. Space shooter tutorial.

# Dynamic Difficulty Adjustment for Maximized Engagement in Digital Games

Su Xue  
Electronic Arts, Inc.  
209 Redwood Shores Pkwy  
Redwood City, CA, USA  
[sxue@ea.com](mailto:sxue@ea.com)

Navid Aghdaie  
Electronic Arts, Inc.  
209 Redwood Shores Pkwy  
Redwood City, CA, USA  
[naghdaie@ea.com](mailto:naghdaie@ea.com)

Meng Wu<sup>\*</sup>  
Electronic Arts, Inc.  
209 Redwood Shores Pkwy  
Redwood City, CA, USA  
[mewu@ea.com](mailto:mewu@ea.com)

Kazi A. Zaman  
Electronic Arts, Inc.  
209 Redwood Shores Pkwy  
Redwood City, CA, USA  
[kzaman@ea.com](mailto:kzaman@ea.com)

John Kolen  
Electronic Arts, Inc.  
209 Redwood Shores Pkwy  
Redwood City, CA, USA  
[jkolen@ea.com](mailto:jkolen@ea.com)

## ABSTRACT

Dynamic difficulty adjustment (DDA) is a technique for adaptively changing a game to make it easier or harder. A common paradigm to achieve DDA is through heuristic prediction and intervention, adjusting game difficulty once undesirable player states (e.g., boredom or frustration) are observed. Without quantitative objectives, it is impossible to optimize the strength of intervention and achieve the best effectiveness.

In this paper, we propose a DDA framework with a global optimization objective of maximizing a player's engagement throughout the entire game. Using level-based games as our example, we model a player's progression as a probabilistic graph. Dynamic difficulty reduces to optimizing transition probabilities to maximize a player's stay time in the progression graph. We have successfully developed a system that applies this technique in multiple games by Electronic Arts, Inc., and have observed up to 9% improvement in player engagement with a neutral impact on monetization.

## Keywords

Dynamic difficulty adjustment, player engagement optimization, progression model

## 1. INTRODUCTION

Difficulty is a critical factor in computer games and is a challenging factor to set appropriately. Game developers often use pre-defined curves that manipulate the level difficulty as players advance. Although these difficulty curves

\*The first two authors contributed equally to this paper.

©2017 International World Wide Web Conference Committee (IW3C2), published under Creative Commons CC BY 4.0 License.  
WWW 2017, April 3–7, 2017, Perth, Australia.  
ACM 978-1-4503-4913-0/17/04.  
<http://dx.doi.org/10.1145/3041021.3054170>



are usually defined by experienced designers with strong domain knowledge, they have many problems. First, the diversity among players is large. Players have a wide variety of experiences, skills, learning rates, and playing styles, and will react differently to the same difficulty setting. Second, even for an individual player, one's difficulty preference may also change over time. For example, in a level progression game, a player who loses the first several attempts to one level might feel much less frustrated compared to losing after tens of unsuccessful trials.

In contrast to static difficulty, dynamic difficulty adjustment (DDA) addresses these concerns. Such methods exhibit diversity in the levers that adjust difficulty, but share a common theme: prediction and intervention. DDA predicts a player's future state given current difficulty, and then intervenes if that state is undesirable. The strength of this intervention, however, is both heuristic and greedy. The adjustment might be in the right direction, such as making a level easier for a frustrated player. But how easy should the game be to achieve optimal long term benefit is an open question.

In this paper, we will address these issues by defining dynamic difficulty adjustment within an optimization framework. The global objective within this framework is to maximize a player's engagement throughout the entire game. We first model a player's in-game progression as a probabilistic graph consisting of various player states. When progressing in the game, players move from one state to another. The transition probabilities between states are dependent on game difficulties at these states. From this perspective, maximizing a player's engagement is equivalent to maximizing the number of transitions in the progression graph. This objective reduces to a function of game difficulties at various states solvable by dynamic programming. While we focus on level-based games as the context of this presentation, our DDA framework is generic and can be extended to other genres.

The proposed technique has been successfully deployed by Electronic Arts, Inc (EA). We developed a DDA system within the EA Digital Platform, to which game clients request and receive dynamic difficulty advice in realtime. With A/B experiments, we have observed significant im-

creases in core player engagement metrics while seeing neutral impact on monetization. Last, but not least, our DDA recommendations are also used by game designers to refine the game design. For example, when our service repeatedly recommends easier difficulty for a certain level, the game designer knows to decrease the pre-defined difficulty of that level to satisfy the majority of population.

To sum up, the core contributions of this paper are:

- We propose a DDA framework that maximizes a player’s engagement throughout the entire game.
- We introduce a probabilistic graph that models a player’s in-game progression. With the graph model, we develop an efficient dynamic programming solution to maximize player engagement.
- We describe a real-time DDA system that successfully boosted engagement of multiple mobile games.

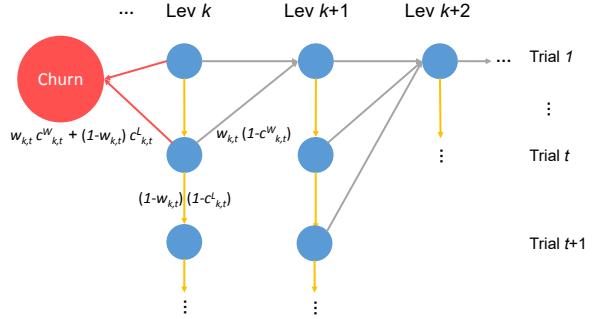
In the remainder of this paper, we will first review related DDA research. We then introduce the graph model of player’s progression and describe the objective function and our optimization solution. We will next report on the application of this DDA technique in a live game as a case study. Finally, we discuss the results of this case study and our future directions.

## 2. RELATED WORK

Personalized gaming is one of the major trends for digital interactive games in recent years [10]. Personalization approaches include content generation, personalized gameplay, and dynamic difficulty adjustment. The theme shared by almost all game difficulty adjustment studies is that they attempt to prevent a player from transitioning to undesired states, such as boredom or frustration. There are several common challenges in difficulty adjustment. For example, how to evaluate a player’s current state? How to predict a player’s upcoming state? What levers are appropriate to use? How to adjust the levers to most appropriate difficulty level? In this section, we review how previous work addressed these questions from different perspectives.

Many approaches are based on the evaluation of players’ skill and performance, and then adapting game difficulty to match the skill level. Zook et al. conducted a series of investigations following this idea [15, 16]. They proposed a data-driven predictive model that accounts for temporal changes in player skills. This predictive model provides a guide for just-in-time procedural content generation and achieves dynamic difficulty adjustment. Similarly, Jennings et al. automatically generate 2D platformer levels in a procedural way [9]. They developed a simulator where players play a short segment of a game for data collection. From this data, they constructed a statistical model of the level element difficulty. They also learned player skill model from the simulator data. Hagelback et al. [6] studied dynamic difficulty by measuring player experience in Real Times Strategy (RTS) games. They, too, use an experimental gaming environment to evaluate testing players’ subjective enjoyment according to different dynamic difficulty schemes.

The majority of DDA systems rely upon prediction and intervention as their fundamental strategy. Hamlet is a well known AI system using Valve’s *Half Life* game engine [8].



**Figure 1: A player’s progression model in a typical level-based game.** We use a probabilistic graph consisting of player states (each circles) to model this progression. Two dimensions, levels and trials are used to identify different states. The directional edges represent possible transition between these states.

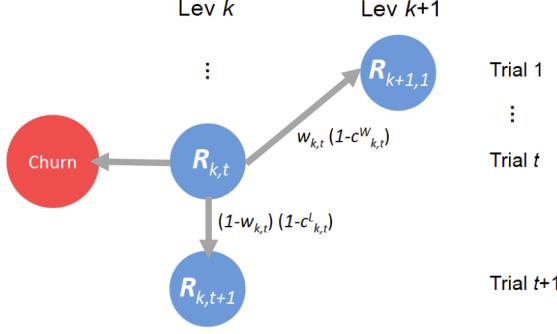
It takes advantages of the flow model developed by Csikszentmihalyi [3], that defines player states on two dimensions: skill and challenge. They suggested the game challenge should match the player skill, therefore some states are preferable while others are not. Hamlet predicts player states, and adjusts the game to prevent inventory shortfalls. Missura and Gartner [11] formalizes the prediction in a more rigorous probabilistic framework. They try to predict the “right” difficulty by formulating the task as an online learning problem on partially ordered sets. Hawkins et al. [7] takes players’ risk profile into account when predicting player performance. Cautious players and risk takers also behave differently in response to dynamic balancing mechanics.

Researchers have also explored the use of various levers to achieve dynamic difficulty. It is preferred that the adjustment by the levers be invisible to players so that they do not feel coddled or abused. Popular levers in the DDA literature include procedural level content [9, 15, 16], off stage elements (such as weapon or support inventory) [8], and AI assistant or opponent [4, 13].

Almost all previous work shares a common limitation. These approaches focus on short-term effects, i.e., using the difficulty adjustment to immediately rescue player from undesired states. With the prediction and intervention strategy, these methods tend to perform greedy actions, often leading to short-term benefits, but failing to achieve long-term optimal impacts. In contrast, our proposed technique achieves DDA by maximizing a long-term objective, such as player’s engagement throughout the entire game. In the following section, we describe how to model the entire game engagement, achieve global optimality, and keep players in the game.

## 3. PLAYER PROGRESSION MODEL

We focus on level-based games in this paper. In a level-based game, a player can unlock and advance to the higher levels only if the player wins the current level. There are many well known digital games e.g. Super Mario Bros. (Nintendo Co., Ltd.) and Plants vs. Zombies (Electronic Arts, Inc.) belong to the level-based game category. We first introduce a progression graph to model players’ progression



**Figure 2: A zoom-in look of the state transitions and the associated rewards.** The reward at  $s_{k,t}$ , i.e.,  $R_{k,t}$ , is the weighted sum of the awards at the adjacent states. This property leads to reward maximization with dynamic programming.

trajectories in the game. The modeling approach described below can be generalized to other game types as well.

Defining appropriate states is the key to constructing a progression model. Specifically for level-based games, we simply define the player progression state with two dimensions, level and trial. A player can either advance to a higher level or remain on the current level with repeated trials. Fig. 1 illustrates the progression graph schema. We denote state that a player is playing the  $k$ -th level at the  $t$ -th trial as  $s_{k,t}$ . Within the progression graph, a player’ progression can be represented by a sequence of transitions between states. For example, when a player completes one level, he will advance to a new first trial state in the next level. When a player fails and retries the same level, he will move to the next trial state on the same level. A special, but critical, state is the churn state. Players who enter the churn state will never return to the game. Hence, the churn state is an absorbing state avoided by the optimization process of DDA.

We now define the transitions between states (represented as directional edges in Fig. 1). A player can only transit to one of two adjacent live states from current live state: 1) the player wins and advances to the first trial of the next level, i.e.,  $s_{k,t} \rightarrow s_{k+1,1}$ ; 2) loses but retries the current level, i.e.,  $s_{k,t} \rightarrow s_{k,t+1}$ . Technically, the assumption is not always true since players are able to retry the current level or play even lower levels after winning. Level replay rarely happens in most games, however. In addition to the transitions described above, all live states can directly transit to the churn state as player leave the game.

Given this structure, we need a probabilistic model of each transition that measures the likelihood of the transition happening. All outgoing transition probabilities sum to one. Since there are only three types of transitions, we can easily investigate each transition respectively.

**Level-up Transition** Starting at state  $s_{k,t}$ , players can level up to state  $s_{k+1,1}$  only if they win and do not churn. Denoting the win rate (i.e., probability to win this level at this state) as  $w_{k,t}$ , and the churn rate after winning as  $c^W_{k,t}$ , we have the level-up probability as:

$$\Pr(s_{k+1,1}|s_{k,t}) = w_{k,t}(1 - c^W_{k,t}) \quad (1)$$

**Retry Transition** From  $s_{k,t}$ , players transits to retrial state  $s_{k,t+1}$  only if they lose and do not churn. The probability of loss is  $1 - w_{k,t}$ . Denoting the churn rate after losing as  $c^L_{k,t}$ , we have the retry probability as:

$$\Pr(s_{k,t+1}|s_{k,t}) = (1 - w_{k,t})(1 - c^L_{k,t}) \quad (2)$$

**Churn Transition** Unless players make the above two transitions, they will churn. The total churn probability at  $s_{k,t}$  is the sum of the churn probabilities after winning and after losing, i.e.,

$$\Pr(churn|s_{k,t}) = w_{k,t}c^W_{k,t} + (1 - w_{k,t})c^L_{k,t} \quad (3)$$

We illustrate the transition probabilities for a given state model in Fig. 1. This probabilistic graph model is the foundation of our optimization framework for dynamic difficulty adjustment in the next section. Note that we assume  $c^W_{k,t}$  and  $c^L_{k,t}$  are independent of  $w_{k,t}$ .

## 4. ENGAGEMENT OPTIMIZATION

### 4.1 Objective Function

With the player progression model, good game design and difficulty adjustment should seek to prevent players from falling into the churn state. DDA achieves higher engagement by adjusting win rates so that the player stays on a state trajectory with lower churn rates. While existing DDA techniques adapt difficulties at each state in a greedy and heuristic manner, our framework identifies optimal win rates for all states, targeting a global objective: maximizing a player’s total engagement throughout the entire game.

Engagement indicates the amount of players’ gameplay. There are multiple engagement metrics, e.g., the number of rounds played, gameplay duration and session days. In this paper, we chose to optimize the total number of rounds played. Three reasons drive this selection. First, the number of rounds a player plays is easily measured in the progression graph. It is the transition count before reaching the churn state or completing the game. Second, many engagement metrics turn out to strongly correlate with each other. We will discuss this observation in Section 5.4. Third, maximizing the number of rounds prevents degenerate solutions that rush a player to the completion state by making the game too easy. Any solution with round repetition will score higher than the shortest path through the graph.

We use  $R$  to denote the reward function, i.e., the expected total number of rounds a player will play through the entire game. While  $R$  hardly looks tractable, we convert it to a more solvable iterative form with the help of the Markov property of the progression graph model. We define reward  $R_{k,t}$  as the expected total number of rounds played after the state  $s_{k,t}$  (level  $k$  with trial  $t$ ). Although we only consider the expectation of the reward in this paper, one could also optimize the variance.

As the player can only transit to two adjacent live states,  $s_{k+1,t}$  and  $s_{k,t+1}$ , or churn,  $R_{k,t}$  can be computed as the weighted sum of  $R_{k+1,t}$  and  $R_{k,t+1}$ . The weights are the transition probabilities between the states. Mathematically, it is written as

$$R_{k,t} = \Pr(s_{k+1,t}|s_{k,t}) \cdot R_{k+1,t} + \Pr(s_{k,t+1}|s_{k,t}) \cdot R_{k,t+1} + 1, \quad (4)$$

where  $\Pr(s_{k+1,t}|s_{k,t})$  is the probability that the player wins and levels up, and  $\Pr(s_{k,t+1}|s_{k,t})$  is the probability that one failed and retries. Adding one represents the reward by completing that round. Transition to the churn state does not contribute to the engagement.

Furthermore, substituting the transition probabilities from Eqns. 1 and 3 into Eqn. 4 (see Fig. 2), produces

$$R_{k,t} = w_{k,t}(1 - c_{k,t}^W) \cdot R_{k+1,t} + (1 - w_{k,t})(1 - c_{k,t}^L) \cdot R_{k,t+1} + 1. \quad (5)$$

Note that the original optimization objective,  $R$ , corresponding to  $R_{1,1}$ . Based on Eqn. 5,  $R_{1,1}$  is a function of win rates at all states,  $\{w_{k,t}\}$ , where  $\{c_{k,t}^W\}$  and  $\{c_{k,t}^L\}$  are parameters that can be extracted from performance data (see details in Section 4.3). Dynamic difficulty adjustment reduces to solving optimal  $\{w_{k,t}\}$  for maximizing  $R_{1,1}$ .

## 4.2 Solving for Optimal Difficulties

We need to solve an optimization problem that finds a set of optimal difficulties over all states, thus

$$\mathcal{W}^* = \arg \max_{\mathcal{W}} R_{1,1}(\mathcal{W}), \quad (6)$$

where  $\mathcal{W} = \{w_{k,t}\}$ . In practice, each  $w_{k,t}$  is constrained by game design and content. We solve for optimal  $\mathcal{W}$  under the constraint that  $w_{k,t} \in [w_{k,t}^{\text{low}}, w_{k,t}^{\text{up}}]$ .

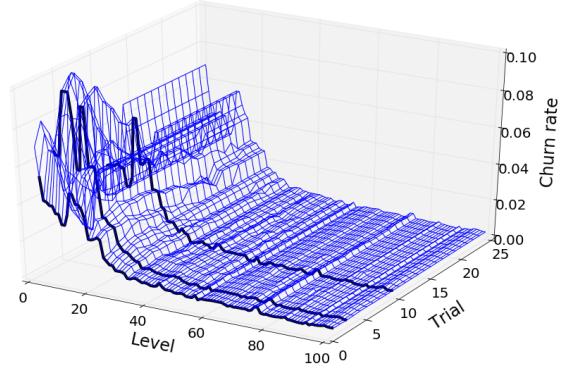
With Eqn. 5, we can solve this optimization effectively with dynamic programming. Denoting  $R_{k,t}^*$  as the maximum reward over all possible difficulty settings, we have:

$$\begin{aligned} R_{k,t}^* = \max_{w_{k,t}} & w_{k,t}(1 - c_{k,t}^W) \cdot R_{k+1,t}^* \\ & + (1 - w_{k,t})(1 - c_{k,t}^L) \cdot R_{k,t+1}^* + 1, \\ \text{s.t. } & w_{k,t} \in [w_{k,t}^{\text{low}}, w_{k,t}^{\text{up}}] \end{aligned} \quad (7)$$

We can see that  $R_{k,t}^*$  is a linear function of  $w_{k,t}$  under a constraint. Therefore, the optimal win rate for state  $s_{k,t}$ ,  $w_{k,t}^*$ , can be found by:

$$\begin{aligned} w_{k,t}^* = \arg \max_{w_{k,t}} & w_{k,t}(1 - c_{k,t}^W) \cdot R_{k+1,t}^* \\ & + (1 - w_{k,t})(1 - c_{k,t}^L) \cdot R_{k,t+1}^* + 1 \\ = \arg \max_{w_{k,t}} & w_{k,t}((1 - c_{k,t}^W) \cdot R_{k+1,t}^* - (1 - c_{k,t}^L) \cdot R_{k,t+1}^*) \end{aligned} \quad (8)$$

Eqn. 8 shows that given the maximal rewards of two future states,  $R_{k+1,t}^*$  and  $R_{k,t+1}^*$ , the optimal difficult  $w_{k,t}^*$  can be computed easily. As the player progression model is a directed acyclic graph, we can solve the optimization with dynamic programming. We start with a few destination states whose rewards are pre-defined and then compute the rewards of the previous states backward. The primary destination state is the end of the game,  $s_{K+1,1}$ , where  $K = k_{\max}$  is the highest level of the game. We assign zero to  $R_{K+1,1}^*$  as the reward for completion of the game. Another set of destination states are those when the number of retrials exceeds a limit, i.e.,  $s_{k,T}$  where  $T \geq t_{\max}$ . By having the upper bound of the retrial time, we can keep the progression graph to a reasonable size. This also prevents a player from too many retrials on a level when the optimization produces unrealistic results. In our experiment we set  $t_{\max} = 30$  and  $R_{k,T}^* = 1$ .



**Figure 3:** An example of a churn surface, which consists of  $c_{k,t}^L$  at all states  $s_{k,t}$ . The churn rates at 1st, 3rd, 10th trials for each level are highlighted in black to illustrate how the churn rate evolves as a player’s re-trials increases.

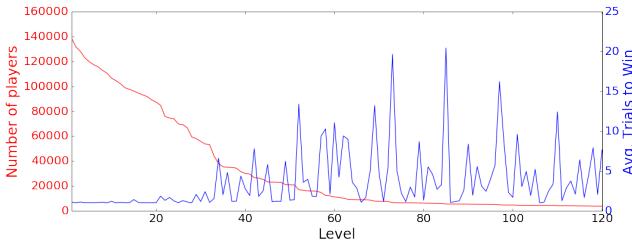
## 4.3 Churn Rates

We assume the churn rates in state transitions ( $c_{k,t}^W$  and  $c_{k,t}^L$ ) as known parameters. In practice, churn is identified as no gameplay during a period of time. We use a 7-day time frame that is common in the game industry and collect historical gameplay data of players at all states (levels and trials) to measure the churn rates. At state  $s_{k,t}$ , let  $c_{k,t}^W$  be the ratio of players who churn after winning, and  $c_{k,t}^L$  after losing. We view the churn rate over all states a two dimensions churn surface. Fig. 3 shows a visual example of a churn surface of  $c_{k,t}^L$ .

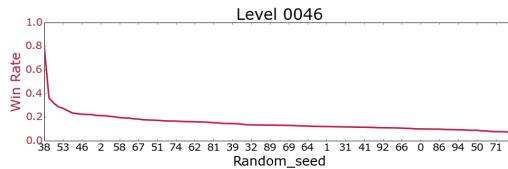
These estimates of churn rates take only two input features, level and trial, while ignoring other players’ individual features such as age, play frequency, play style, skill, performance, etc. To further improve the accuracy of churn rates, we could take advantage of long-standing churn prediction research [2, 5, 12], by employing sophisticated predictive models on individual player features to improve performance. A major challenge of using runtime churn prediction is that it increases the complexity of dynamic programming optimization. Pre-computation with various possible churn rates at each state (via bucketing) would be needed. This mechanism is not employed by our current system, but will be worth exploring in the future.

## 5. CASE STUDY: DDA IN A LIVE GAME

We now present the case study with one of our DDA implementations, a mobile match-three game developed and published by EA. Classic example of match-three genre, e.g. Candy Crush Saga by King and Bejeweled by EA, has a game board contains multiple items in different colors and shapes. A player can swap two adjacent items in each move as long as three or more items of the same color become aligned together vertically or horizontally. The aligned items will be eliminated from the board. At each level, a player needs to achieve a specific goal, for example, score a number of points in a limited number of moves. The game features more than two hundred levels. A player starts from the lowest level and advances to the higher levels. Only if a player wins the current level, the next higher level will be unlocked.



**Figure 4:** The retained population of players (red line) versus difficulties (blue line) by level. The red line represents, for each level, the number of players who have ever achieved it. The blue line represents the level difficulty, which is measured by 1/win rate, i.e., the average trials needed to win this level. We can observe the strong impact of difficulty on population retention, in particular for middle stage levels.

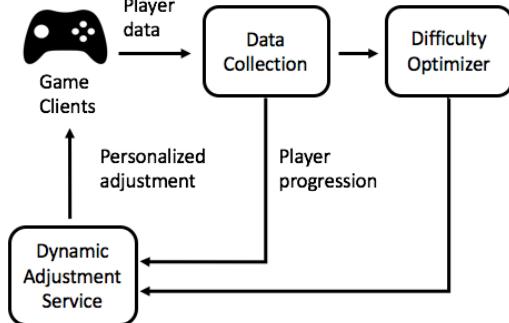


**Figure 5:** Difficulties of various random seeds at a level of the match-three game. Difficulty is measured by the win rate of a certain seed, i.e., the percentage out of all trials with this seed are actually wins. The variance of difficulties across seeds is large. We can see that the easiest seed (leftmost, seed 38) shows a win rate up to 0.75. In contrast, the hardest seed (rightmost, seed 71) has a win rate as low as 0.15.

Before adopting DDA, we must ask: can DDA help this game? To answer this question, we must convince ourselves of a causal link between the difficulty and engagement. First, we should determine if game difficulty is affecting the player engagement in the game. Second, appropriate levers should exist to effectively adjust the game difficulty. We will examine these two prerequisites in the following sections.

## 5.1 Difficulty and Engagement

To evaluate the impact of difficulty on player engagement, we compared the retained population with level difficulties (see Fig. 4). Retained population (red line) at a level is the number of players who have achieved this level as the highest one. There are players churned at each level, thus the retained population decreases as the level increases. The difficulty (blue line) is measured by the average number of trials that are needed to win this level. The more trials it takes, the more difficult this level is. Dividing all levels into three ranges: lower range ( $\leq 20$ ), middle range (21-80) and high range ( $> 80$ ), we can see that the correlation between retained population and difficulty varies. In the lower range of levels, the population naturally decays regardless of the low difficulty of these levels. Especially at level 21, there is a slight increase in difficulty, and the number of players drops significantly. In the high range of levels, the population becomes flat and decays slowly, despite that these high levels are very difficult. In contrast, in the middle range, dif-



**Figure 6:** Schematic diagram of the Dynamic Difficulty Adjustment system.

ficulty spikes are highly correlated with the steep drops in retained population. This observation supports the hypothesis that appropriate difficulty adjustment has the potential to enhance player engagement for this game.

## 5.2 Difficulty Lever

DDA adjusts win rates at different states in the progression graph through a difficulty lever. An effective difficulty lever needs to satisfy two key constraints. First, adjusting this lever should make the game easier or harder. Second, adjusting this lever should be invisible to players (as reviewed in [10]). For example, although we can simply change the “goal” or “mission” to lower game difficulty, players can easily observe it in retrials. As a consequence, such changes undermine the players’ sense of accomplishment even when they finally win with the help of DDA.

Fortunately, the case study game provides an effective difficulty lever: the random seed of board initialization. At the beginning of each round, the board is initialized from a random seed, which is indexed by a integer from 0 to 99. After evaluating the average win rate of each seed in gameplay data, we find a wide range of difficulties. Fig. 5 shows an example of difficulties versus seeds at one level. The seeds are ordered by their observed win ratios. We can see that the easiest seed (leftmost) has a win rate as high as 0.75, while the hardest seed (rightmost) has a win rate as low as 0.15. The player who plays the hardest seeds will take 5x more trials to pass this level than those playing the easiest seeds. This variance can be explained by the game mechanism. For example, some initial boards have many items of the same color close to each other, making it significantly easier to match items than boards with more pathological scattering. By carefully selecting the seed according the mapping in Fig. 5, we can control the game difficulty for players on this level. The hardest and easiest seeds provide the lower and upper bounds of win rates, i.e.,  $w^{low}$  and  $w^{up}$  in Eqn. 6.

## 5.3 Dynamic Adjustment System

We developed a real-time system to serve dynamic difficulty adjustments in an EA match-three game. Fig. 6 describes the workflow of the DDA system. At the beginning of each game round, the game clients send a request to the DDA service. The dynamic adjustment service determines the most suitable difficulty for the current player state,  $s_{k,t}$

based on the player progression and difficulty optimization results,  $w_{k,t}^*$ . The optimal win rates are computed offline as discussed in Section 4.2. Since we use random seeds as the difficulty lever, the dynamic adjustment service then applies the mapping from win rates to the random seeds showed in Fig. 5 and return it to the game client. In practice, we randomly select one seed from the top five candidate seeds to prevent from repeatedly playing only one seed. The game was first released in a period without DDA (soft launch phase), allowing the measurement of win rate for each random seed. After DDA is started, we continued collecting player data to improve the random seed mapping, churn probabilities, and difficulty optimization.

## 5.4 Experimental Results

To measure the effectiveness of our technique, we conducted A/B experiments that use multiple variants as control and treatment groups. The control group randomly recommends seeds out of all possibilities, an action corresponding to the default game behavior. The treatment group recommends the seeds based on our DDA optimization. We calculated all parameters for optimization, such as the churn surface and win rates of seeds, from real game play data. We kept track of these parameters and updated them when significant changes were observed.

The experiment started two weeks after the global release of the game. We conducted the experiment in three phases, where the proportion of the treatment group increased gradually from 10% to 40%, and finally 70% (the proportion of the control group decreases from 90%, 60% to 30%). The first two phases each lasted about one month. The third phase has been live for about four months and is ongoing. We compared core engagement metrics between the control and the treatment groups to evaluate the effectiveness of our DDA scheme. The results are daily averages normalized according to the proportion of its group, so that groups with different proportions could be compared. Phase III has not been terminated in order to collect churn probabilities and evaluate performance.

Table 1 shows the increase of the number of rounds played, suggesting that the DDA is optimizing its objective metric. In each phase of increasing treatment populations, all treatment groups exhibits statistically significant improvement ( $p\text{-value} < 0.05$ ). Table 2 shows the impact of DDA on another engagement metric, total gameplay duration. Though this metric is not the explicit optimization objective of DDA, we wanted to test an alternative hypothesis that players played more rounds in the same amount of time. Our DDA treatment group shows significant improvement on this engagement metric as well. Similarly, performance increased as more data was collected in the second phase; then stayed stationary when the gameplay data became accurate and stable in the latest phase. Note that we see slightly different performances between iOS and Android platforms, though, in the same order. This resonates with the common observation in mobile game development that user behaviors between platforms often differ from each other [1, 14], so that separate modeling and treatment are preferred.

We observed the lowest performance in Phase I, when DDA is completely based on the model we learned from limited data - soft launch data and first two weeks in worldwide release. The soft launch data is only partially useful as some game design is changed before worldwide release. After

| Phase | Platform | Default   | DDA       | Delta   | Gain  |
|-------|----------|-----------|-----------|---------|-------|
| I     | iOS      | 1,118,237 | 1,167,616 | +49,379 | +4.4% |
|       | Android  | 789,640   | 825,182   | +35,543 | +4.5% |
| II    | iOS      | 855,267   | 915,334   | +60,067 | +7.0% |
|       | Android  | 1,137,479 | 1,228,473 | +90,995 | +7.9% |
| III   | iOS      | 711,749   | 763,508   | +51,759 | +7.2% |
|       | Android  | 1,285,448 | 1,366,820 | +81,373 | +6.3% |

**Table 1:** Total numbers of rounds played daily in the default (control) group versus in the DDA treated group. Delta is the absolute increase by DDA and Gain is the relative increase.

| Phase | Platform | Default   | DDA       | Delta    | Gain  |
|-------|----------|-----------|-----------|----------|-------|
| I     | iOS      | 3,684,082 | 3,847,516 | +163,435 | +4.4% |
|       | Android  | 2,686,781 | 2,814,953 | +128,172 | +4.8% |
| II    | iOS      | 2,916,570 | 3,148,722 | +232,152 | +7.9% |
|       | Android  | 3,787,414 | 4,129,762 | +342,348 | +9.0% |
| III   | iOS      | 2,582,809 | 2,788,690 | +205,881 | +8.0% |
|       | Android  | 4,619,907 | 4,956,680 | +336,773 | +7.3% |

**Table 2:** Total durations of gameplay time (in minutes) daily in the control group versus in the DDA treated group.

the release, we continued to collect more data to form more accurate parameters for DDA optimization. This explains the further improved performance in Phase II over Phase I. In Phase III, the performance has been observed stable for more than three months. The amount of boost is relatively smaller than that of Phase II because there are fewer new players in this phase and DDA has higher impacts on early stage players (see Fig. 4).

Last but not least, we also compared the impact on monetization between the control and the treatment groups. This comparison is critical as a monetization objective might contradict engagement maximization. Our DDA treatment group had a neutral impact on monetization. No statistically significant difference on in-game transaction revenues was observed between two groups. This is probably caused by the mechanism of our optimization framework: it “saves” players of high churn risks, who are less likely to spend.

## 6. CONCLUSION

In this paper, we presented a framework that approaches dynamic difficulty adjustment (DDA) as an optimization problem. While existing DDA systems adapt game difficulty in a greedy manner for local benefit, our method maximizes the player engagement throughout the entire game. We modeled the player progression as a probabilistic graph, with which engagement maximization becomes a well-defined objective and we proposed an efficient dynamic programming method to solve it. We evaluated an implementation of the DDA technique in a mobile game by EA. The DDA treated group shows significant increases in core engagement metrics, such as a total number of rounds played and gameplay duration, while it is revenue neutral compared to the control group with no DDA enabled.

In the near future, we will extend the framework to a wide range of game genres. The key to successfully applying DDA to other genres is the construction of an adequate progression model. For level-based games, the states can be naturally identified by two major dimensions: level and trial. For more complicated games with non-linear or multi-

ple progressions, such as role-playing games (RPG), we can also define the states with different dimensions. The graph might be more complicated as it includes more states and connections. Solving the optimal difficulty associated with each state, which maximizes player engagement, however, remains a well-defined graph optimization problem.

Another problem worth investigating is the cold-start stage when the graph parameters (such as seed difficulty and churn risks) needed for optimization are not known yet. In the current system, we learned these parameters in soft launch and the first few weeks after release before starting treatment (see Section 5.4). A possible direction is to conduct reinforcement learning at this early state, by defining some short-term awards that help quickly establish greedy policies. Once sufficient data is collected, we can then shift to the supervised optimization framework.

Last but not least, we would like to explore generic dynamic player experience. Beyond difficulty adjustment, how can we provide a dynamic and intelligent game experience personalized for individual players? It includes, but not limited to, dynamic tutorials, dynamic messages, dynamic awards, dynamic advertisement and etc. We believe that the progression model and corresponding optimization framework will become key to these generic solutions.

## 7. REFERENCES

- [1] Z. Benenson, F. Gassmann, and L. Reinfelder. Android and ios users' differences concerning security and privacy. In *CHI '13 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '13, pages 817–822, New York, NY, USA, 2013. ACM.
- [2] R. Bernhaupt. User experience evaluation in entertainment. In *Evaluating User Experience in Games*, pages 3–7. Springer, 2010.
- [3] M. Csikszentmihalyi and I. S. Csikszentmihalyi. *Optimal experience: Psychological studies of flow in consciousness*. Cambridge university press, 1992.
- [4] P. Demasi and J. d. O. Adriano. On-line coevolution for action games. *International Journal of Intelligent Games & Simulation*, 2(2), 2003.
- [5] F. Hadjii, R. Sifa, A. Drachen, C. Thurau, K. Kersting, and C. Bauckhage. Predicting player churn in the wild. In *2014 IEEE Conference on Computational Intelligence and Games*, pages 1–8. IEEE, 2014.
- [6] J. Hagelback and S. J. Johansson. Measuring player experience on runtime dynamic difficulty scaling in an rts game. In *2009 IEEE Symposium on Computational Intelligence and Games*, pages 46–52. IEEE, 2009.
- [7] G. Hawkins, K. Nesbitt, and S. Brown. Dynamic difficulty balancing for cautious players and risk takers. *Int. J. Comput. Games Technol.*, 2012:3:3–3:3, Jan. 2012.
- [8] R. Hunicke. The case for dynamic difficulty adjustment in games. In *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology*, pages 429–433. ACM, 2005.
- [9] M. Jennings-Teats, G. Smith, and N. Wardrip-Fruin. Polymorph: dynamic difficulty adjustment through level generation. In *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*, page 11. ACM, 2010.
- [10] S. Karpinskyj, F. Zambetta, and L. Cavedon. Video game personalisation techniques: A comprehensive survey. *Entertainment Computing*, 5(4):211–218, 2014.
- [11] O. Missura and T. Gärtner. Predicting dynamic difficulty. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 2007–2015. Curran Associates, Inc., 2011.
- [12] J. Runge, P. Gao, F. Garcin, and B. Faltings. Churn prediction for high-value players in casual social games. In *2014 IEEE Conference on Computational Intelligence and Games*, pages 1–8. IEEE, 2014.
- [13] P. Spronck, I. Sprinkhuizen-Kuyper, and E. Postma. Difficulty scaling of game AI. In *Proceedings of the 5th International Conference on Intelligent Games and Simulation (GAME-ON 2004)*, pages 33–37, 2004.
- [14] C. Zhou, Y. Guo, Y. Chen, X. Nie, and W. Zhu. Characterizing user watching behavior and video quality in mobile devices. In *2014 23rd International Conference on Computer Communication and Networks (ICCCN)*, pages 1–6, Aug 2014.
- [15] A. Zook, S. Lee-Urban, M. R. Drinkwater, and M. O. Riedl. Skill-based mission generation: A data-driven temporal player modeling approach. In *Proceedings of the The Third Workshop on Procedural Content Generation in Games*, PCG'12, pages 6:1–6:8, New York, NY, USA, 2012. ACM.
- [16] A. Zook and M. O. Riedl. A temporal data-driven player model for dynamic difficulty adjustment. In *Artificial Intelligence and Interactive Digital Entertainment*, 2012.

# Enemy Within: Long-term Motivation Effects of Deep Player Behavior Models for Dynamic Difficulty Adjustment

Johannes Pfau

Digital Media Lab, TZI,  
University of Bremen  
Bremen, Germany  
jpfau@tzi.de

Jan David Smeddinck

Open Lab, School of Comp.,  
Newcastle University  
Newcastle upon Tyne, UK  
jan.smeddinck@newcastle.ac.uk

Rainer Malaka

Digital Media Lab, TZI,  
University of Bremen  
Bremen, Germany  
malaka@tzi.de

## ABSTRACT

Balancing games and producing content that remains interesting and challenging is a major cost factor in the design and maintenance of games. Dynamic difficulty adjustment (DDA) can successfully tune challenge levels to player abilities, but when implemented with classic heuristic parameter tuning (HPT) often turns out to be very noticeable, e.g. as “rubberbanding”. Deep learning techniques can be employed for deep player behavior modeling (DPBM), enabling more complex adaptivity, but effects over frequent and longer-lasting game engagements, as well as comparisons to HPT have not been empirically investigated. We present a situated study of the effects of DDA via DPBM as compared to HPT on intrinsic motivation, perceived challenge and player motivation in a real-world MMORPG. The results indicate that DPBM can lead to significant improvements in intrinsic motivation and players prefer game experience episodes featuring DPBM over experience episodes with classic difficulty management.

## CCS Concepts

- Human-centered computing → User models;
- Computing methodologies → Neural networks;
- Applied computing → Computer games;

## Author Keywords

Dynamic difficulty adjustment; Player Modeling; Neural Networks; Deep Learning; MMORPGs; Games

## INTRODUCTION

With the ongoing rise of complexity, popularity and content production cost of video game development, consistently balanced challenges that keep players motivated over the long term are becoming hard to attain, especially with large player communities encompassing broad ranges of proficiency. Dynamic Difficulty Adjustment (DDA) [23] denotes the principle of adapting video game challenges to players’ abilities - both mental and physical / dexterity - in order to allow motivation-fostering flow states [10] to arise. It has been successfully

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

CHI ’20, April 25–30, 2020, Honolulu, HI, USA.

© 2020 Copyright is held by the owner/author(s). Publication rights licensed to ACM. ACM ISBN 978-1-4503-6708-0/20/04 ..\$15.00.  
<http://dx.doi.org/10.1145/3313831.3376423>

deployed in scientific [23, 24, 46] and industrial [7, 16, 55] contexts and is usually accomplished by continuous tuning of core game variables (such as speed, damage or hit ratio). However, these systems are inherently limited to a small number of high-level parameters, require careful tuning of thresholds in heuristic parameter tuning (HPT) [51] and have to be hidden to avoid exploitation; e.g. as not to incentivize players to perform badly on purpose [43]. This results in limited expressivity and complexity of system behavior, as well as in considerable development cost. To address these limitations of classic HPT, we present a novel DDA strategy by implicitly assessing individual player proficiency using Deep Player Behavior Modeling (DPBM) [38] and generating adaptive, personalized challenges. Player behavior, in terms of state-action decision making, is captured while fighting an in-game opponent and trained onto an individual, initially randomized model. Upon the next encounter, the opponent uses this model generatively by retrieving action probabilities for each game state emerging in an interaction. As a consequence, its decision making approximates the original player’s behavior, implicitly representing the particular game proficiency. Evaluating the real-world applicability of DPBM, we aim to answer the following research questions:

- Do players perceive behavior from DPBM as representative of their own decision making?
- Is the players’ self-reported intrinsic motivation when interacting with a DPBM opponent higher than for traditional HPT encounters?
- Can we measure a substantial long-term motivation achieved by DPBM?

We hypothesize that an agent that keeps up with the progress of the player, displays similar weaknesses and challenges players to constantly improve or rethink strategies will yield a novel and captivating take on DDA. For the purpose of evaluating the differences between HPT and DPBM, we implemented *Eternal Challenge* – an adaptive instance dungeon inside of the popular Massively Multiplayer Online Role-Playing Game (MMORPG) *Aion* [34] – and assessed players’ motivation in a field study ( $n = 171$ ) on an existing private server. After a deployment of four weeks, we were successful in showing a significantly higher long-term usage of the instance compared to all alternatives and that the DPBM opponent contributes significantly more to this motivation than HPT mechanics,



Figure 1. Appearance of the different opponents with DDA through HPT variables and DPBM.

based on quantitative – and supported by qualitative – insights. Furthermore, some players stated that they noted the progress of the DPBM opponent in learning their own individual strategies, leading to a unique game experience. We contribute to games user research and inform game development by investigating DPBM as a form of novel DDA in a situated medium- to long-term study, showcasing its distinct potential for fostering intrinsic motivation and demonstrating a working approach with learning adaptive opponents in the wild.

## RELATED WORK

Providing and balancing an accurate level of difficulty is critical for keeping players constantly engaged [2, 23]. Disparities can ultimately lead to boredom/underload or frustration/overload, which make for two of the main causes why players stop playing games [11]. Since individual skill and its progression are hard to foresee throughout potentially large player bases and difficulty and its progression can not be defined or programmed precisely, the field of DDA attempts to regulate emergent mismatches dynamically. To estimate imbalanced challenge-proficiency-discrepancies, various assessment techniques have been researched, such as success probability estimation [24, 32] or biofeedback [22, 25, 35]. However, When it comes to adjusting this difficulty, most approaches focus on HPT (apart from procedural world generation [26, 57, 50]), even in the most recent advancements [3, 4, 8, 17, 18, 40].

Opponents that imitate the player character exist in numerous commercial games, perhaps most notably the recurring *Dark Link* in the *The Legend of Zelda* series [15], the *Guild Wars Doppelganger* [5], *Renegade Shepard* from *Mass Effect 3* [6] and *SA-X* in *Metroid Fusion* [42]. Yet, so far these have only been realized as crude approximations of the original player, as they mimic appearance, equipment, basic moves and/or skill sets by relying on heuristic, strategically rigid decision-making.

At the same time, machine learning approaches in video games that harness continuous improvement through simulated play have become popular, e.g. the deep reinforcement learning of *Atari* games [31], temporal difference learning of *Backgammon* [54] or the surpassing of human player performance in the board game that has been rated as not solvable by artificial intelligence methods for a long time; *Go* [48]. The field of player modeling has seen explorations of machine learning techniques for multiple purposes, prominently featuring

prediction, classification or analysis [13, 14, 27, 53], to facilitate individual game interpretations, testing or for providing believable opponents. Still, the incorporation of machine learning approaches for generative player modeling for DDA and the resulting player experiences remain under-investigated. Holmgård et al. studied *personas* for player decision modeling [19, 20] that continually observe and adapt to human behavior in order to produce agents with different decision-making styles. These *personas* were realized via evolutionary linear perceptrons and compared to heuristic agents in a test bed 2D dungeon crawler game, resulting in a higher player-rated human-likeness. They also assessed player models when defined as “deviations from theoretically rational actions” in a study of *Super Mario Bros.* [1, 21] and clustered these by means of feature extraction. Using the same game, Ortega et al. [36] imitated human playing styles by means of Neuroevolution and Dynamic Scripting and reached higher scores of human-likeness than performance-directed AI agents, based on subjective judgments. Missura and Gärtner utilized Player Modeling in a 2D test bed shooter via Support Vector Machines acting as predictors for difficulty mismatches and enabling classical DDA parameter tuning based on the results [30]. In previous work, we were successful in showing that player model agents can yield significantly higher motivation compared with heuristic opponents in a short-term online study using the 2D platform fighter *Korona:Nemesis* [9, 39]. Based on these insights, player awareness about substituting individual players with DPBM agents in online multi-player matches was also assessed. In contrast to heuristic bots, DPBM agents turned out to be indistinguishable from their human precursors [37]. In addition, we contrasted different machine learning techniques in a player modeling study of the MMORPG *Lineage 2* [33, 38]. Deep learning offered the best individual prediction accuracy, facilitating the production of playing sessions that closely resemble the original behavior, as well as for differentiating between players. Consequently, we discussed the broader implications for the application of DPBM in: DDA (offering adaptation beyond parameter tuning; training players by exposing them to own strengths and weaknesses), player substitution (bridging online match disruption due to dropouts; providing more individually representative agents), automated game testing (enhancing the estimation of balancing issues by incorporating realistic human player behavior) and cheating detection (revealing behavior that is more likely to stem from undesirable third-party bots rather than players; yield-

ing objective evidence based on behavior in cases of identity theft).

To the best of our knowledge, there is no prior work assessing the experience of players who continuously challenge themselves, where generative player modeling facilitates proficiency progress.

## APPROACH

In contrast to the aforementioned studies, the approach presented in this work provides a medium- to long-term situated evaluation. We compare the deployment of player modeling through DPBM with traditional HPT and assess feasibility, approval and motivation in a complex AAA game through a highly ecologically valid field study. To facilitate realistic and generalizable results that avoid artificial laboratory study setting biases [44], we aimed for the deployment of our approach in a real-world setting in a fully fledged game with an existing community of players. In the following, we explain the construction of the recorded training data format, how it was informed by expert interviews, the DPBM architecture, and the study environment.

### Expert Interview

In order to gain a more elaborate understanding of viable strategies, decision making and what behavior might lead to different play styles in *Aion*, one of the authors consulted 3 expert players of the game (each with 7–8 years of prior experience) and extracted the most important aspects qualitatively, using brief 1 hour semi-structured interviews over the course of one day. Apart from a less-structured introduction and follow-up discussion after each item, we asked the following questions:

- In a one-on-one situation against a (computer controlled opponent / human player), based on which factors do you decide which skill to use?
- How do you react when you are not able to execute your strategy?
- How would you approach an opponent of the same class that is equally proficient as yourself?

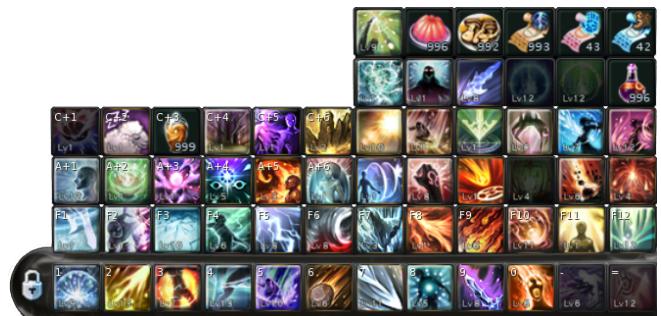
We analyzed the interview using an outcome-oriented structuring content analysis after Mayring [28] and consolidated the most expressive statements about factors that qualify as good indicators for decision making. The most descriptive factors as indicated by the experts are adherence to skill *rotations* and situational *responsive decisions*. Within *rotations*, expert players predominantly apply a specific set of preferred sequences of actions, e.g. ramping up damage by combinations of enhancing and weakening skills or controlling the opponent by a succession of restricting skills. Due to the large amount of possible skills or items to use (cf. Figure 2), these rotations can include complex chains of consecutive skills and/or contain *sub-rotations*. *Responsive decisions* denote the reaction to certain states that the player character or an opponent is in, e.g. healing oneself when hit points are low, removing restricting conditions on the character, increasing or reducing distance between characters or exploiting temporary

conditions the opponent is in. They can also trigger more complex situation-specific *rotations*. The description of play-style aspects by means of *rotations* and *responsive decisions* is not limited to this specific game, but broadly generalizes to the genre of action RPG games. We defined the DPBM state-action architecture on the basis of these factors, including player and target state information as crucial indicators for *responsive decisions* and previous skill information for positioning within *rotations*.

In combination, these factors compose the game state that is fed into the DPBM input layer, while the output layer is trained according to the respective skill that the player used in this situation (cf. Figure 3).

### Adaptive Instance Dungeon

Instance dungeons are a major part of MMORPGs, as they can be entered numerous times, solo or in a group, to acquire experience points, equipment, currencies and/or other desirable items. As such, they provide a fertile testing ground for evaluating long-term motivation [52], since most often repeated or even continuous entries are required to reach higher-level goals. To gather expressive evidence of the motivational potential of DPBM, we developed the single-player adaptive instance dungeon *Eternal Challenge* that incorporates both traditional DDA aspects via HPT as well as DPBM. Within the instance, the players encountered various opponents that were clearly distinguishable by their visual appearance (cf. Figure 1) and were adjusted through distinct parameters tuned by HPT (cf. Table 1). The underlying proficiency variable  $\lambda$  approximated the player's performance level by being increased whenever he successfully completed *Eternal Challenge* and decreased at the characters death or temporal expiry of the countdown. This way, HPT produces a typical “rubberbanding” effect between player-specific thresholds of lack of challenge and excessive challenge, which is one of the most common ways to enable flow-states to arise in traditional DDA [47] and was constructed by following the inspiration of these approaches [17, 23, 30, 47] in combination with fine-tuning by the developers operating the server to find a range covering too easy, too hard and enough configurability in between for every observed player.



**Figure 2.** Exemplary arrangement of a subset of skills available to the Sorcerer class in *Aion*. Additionally, context-dependent skills (when the player or a target opponent is in a particular condition) and a multitude of items can be activated.

|                            |  |
|----------------------------|--|
| $\lambda$ difficulty level | Increased whenever <i>Eternal Challenge</i> was completed successfully, decreased upon death or timeout.   |
| $\alpha$ frequency         | With increased $\lambda$ , the temporal spawn delay of $\alpha$ opponents was decreased, resulting in an exponential increase of difficulty.                         |
| $\beta$ perseverance       | With increased $\lambda$ , hit points (HP) of $\beta$ opponents increased, making them harder / more time-consuming to defeat.                                       |
| $\gamma$ disturbance       | With increased $\lambda$ , $\gamma$ opponents used more actions that weaken the player, which decreases survivability, damage performance and increases the tension. |
| DPBM                       | No explicit parameter tuning was used, since network proficiency approximates the player's skill implicitly.   |

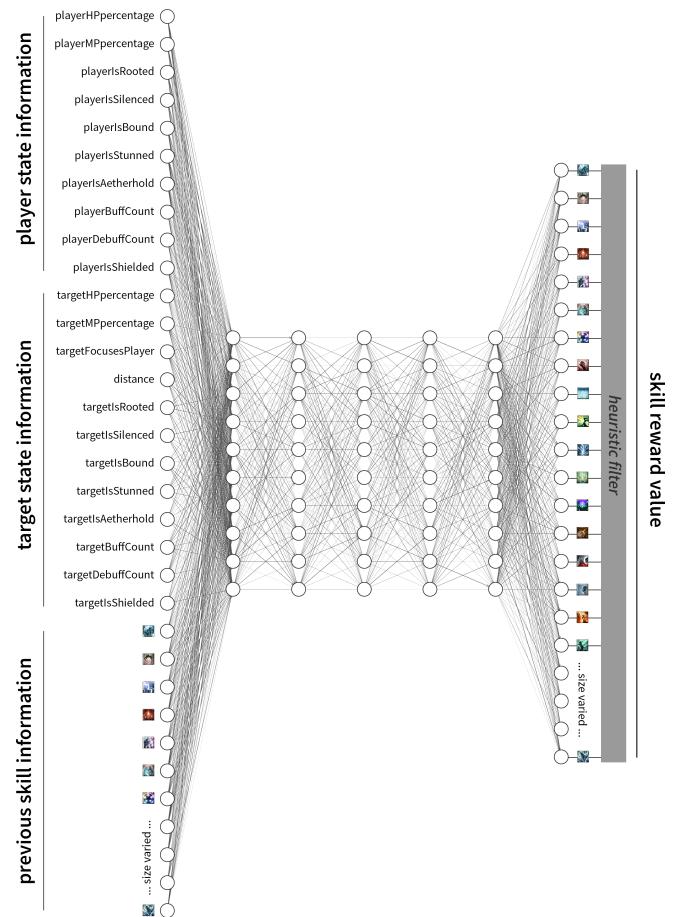
**Table 1. DDA mechanisms of *Eternal Challenge*, mapping  $\lambda$  to the difficulty of  $\alpha, \beta, \gamma$ , while DPBM seeks to emulate the player's behavior.**

To avoid incentivizing players to perform badly on purpose, rewards (in the form of experience gained and the level-range of items dropped) were adjusted to be proportional to the difficulty level  $\lambda$ . Upon entering the instance, a 15-minute countdown started that expelled the player if it was not finished after expiration. Within this time limit, the player was expected to destroy a sturdy, non-responsive opponent ( $\beta$ ) that spawned additional, hostile enemies over time ( $\alpha, \beta, \gamma$ ) which had to be endured or defeated as well. As soon as the main opponent was defeated, an additional foe that utilizes DPBM appeared in an adjacent room. If the player managed to beat this opponent as well, rewards were distributed and the internal  $\lambda$  level was raised accordingly.  $\lambda$  had no theoretical, but a practical upper limit, since the game inherently restricts reaching damage per second values beyond a certain threshold.

### Deep Player Behavior Modeling

When entering *Eternal challenge*, the recorded behavioral data from all preceding runs of the player was retrieved from the underlying database and fed into a feed-forward multi-layer perceptron with backpropagation and a logistic sigmoid activation function (cf. Figure 3), where input and output layer size varied depending on the player's class, skill set and usage ( $M = 98.2, SD = 15.1$ ) input,  $5 \times 10$  hidden, ( $M = 76.2, SD = 15.1$ ) output nodes). The network was initialized randomly and trained over 1000 epochs, based on the insights of previous work [38, 39] and benchmarks prior to the study that indicated diminishing returns when further increasing the range of parameters. When encountering the DPBM opponent, the trained model was applied generatively to retrieve a set of action probabilities given the occurring state description at real-time. After a weighted choice, the resulting skill was executed, followed by querying the DPBM

for the next situation, effectively approximating the learned behavior from the player's preceding battles. As movement was controlled heuristically, temporal-dynamics of behavior are not explicitly modeled, but behavior over time is modeled by focusing the sequencing of skill *rotations* and *responsive decisions* in each occurring state. In terms of the player modeling taxonomy of Yannakakis et al. [58], this implementation realizes a *model-free* (bottom-up) player modeling approach mapping *gameplay data* to actions via *preference learning* and *classification*. According to the player modeling description framework of Smith et al. [49], DPBM directly utilizes *game actions* (**domain**) to generate (**purpose**) individually (**scope**) modeled behavior by means of induced (**source**) training of machine learning techniques.



**Figure 3. The DPBM architecture mapping game state (information about player, opponent and skill history) to action (skill usage) probabilities. The size of the input and output layers varied depending on the player's class, skill set and usage. The resulting action probability array is filtered heuristically by removing skills that are impossible to execute due to cool-down, MP shortage or other insufficient conditions.**

## STUDY

In order to evaluate the DDA and intrinsic motivation capabilities of DPBM, we conducted an online field study following a within-subjects design over the course of four weeks. The adaptive instance was published on a private *Aion* game server. To ensure that the measured motivation is attributable to the DPBM approach, we took several precautions. In order to minimize novelty or anticipation bias, we did not announce the existence or concept of the instance prior release and chose a long-term study design. In addition, rewards constitute one of the biggest extrinsic motivators for long-term commitments [56] and thus a potentially high confounding factor when assessing intrinsic motivation. Therefore, the rewards of *Eternal Challenge* were kept conservative and approximated the amount of rewards in other available instances, i.e. players were not able to obtain something that they would not be able to elsewhere and did not acquire a higher reward-to-time-ratio. Findings of Deci et al. [12] also suggest that excessive extrinsic rewards can inhibit intrinsic motivation, the core factor of engagement and enjoyment [45, 41]. Finally, as interplay among players is a major motivating factor of MMORPGs [56], we excluded multi-player situations, leaderboards, high score lists or the publication of ranks and completion times during the study period to avoid complex potential biases in this early situated study. Although MMORPGs are designed to be about multi-player scenarios, occasions of playing solo occur on a regular basis and novel challenge paradigms should arguably be tested in basic, controllable setups before being extended to include additional factors, such as team play or competition.

## Measures

For every single *Eternal Challenge* run performed by a player, we recorded state-action data for DPBM training (cf. Figure 3), instance completion times and results (failed or succeeded), as well as the training times and prediction accuracies of the DPBM. In addition, we logged entry counts and timestamps for all available instance dungeons for further activity comparisons. After the study period, a post-study questionnaire asked for player-reported assessments of *perceived competence*, *interest-enjoyment*, *tension-pressure* and *effort-importance*, following the *Intrinsic Motivation Inventory* (IMI) [29], for comparison between the traditional DDA parameters  $\alpha$ ,  $\beta$ ,  $\gamma$ , and the DPBM opponent. Each iteration of the questionnaire was explicitly headed by a display of the appearance of the corresponding opponent in order to assure correctly targeted responses (cf. Figure 1).

Additionally, the survey contained qualitative queries about the appreciation of – and strategies used against – the opponents, the impression of DPBM opponent’s behavior in the players’ own words, and a free field for additional remarks.

## Procedure

The instance dungeon *Eternal Challenge* was introduced and released as part of a regular update to the private server. From then on, players of the community had the opportunity to enter it up to once daily, independent from entering different or additional instances. After four weeks, the recording of in-game data stopped and the post-study questionnaire was advertised on a message board associated with the server.

## Participants

During the study period, ( $n = 171$ ) participants entered *Eternal Challenge* resulting in 776 total instance runs. 30 players (17 men, 13 women (self-identified)) completed the optional post-study questionnaire.

## RESULTS

Using a one-way RM ANOVA, we found significant effects for the IMI scores *perceived competence*, *interest-enjoyment*, *tension-pressure* and *effort-importance* between conditions  $\alpha$ ,  $\beta$ ,  $\gamma$  and DPBM (cf. Table 2).

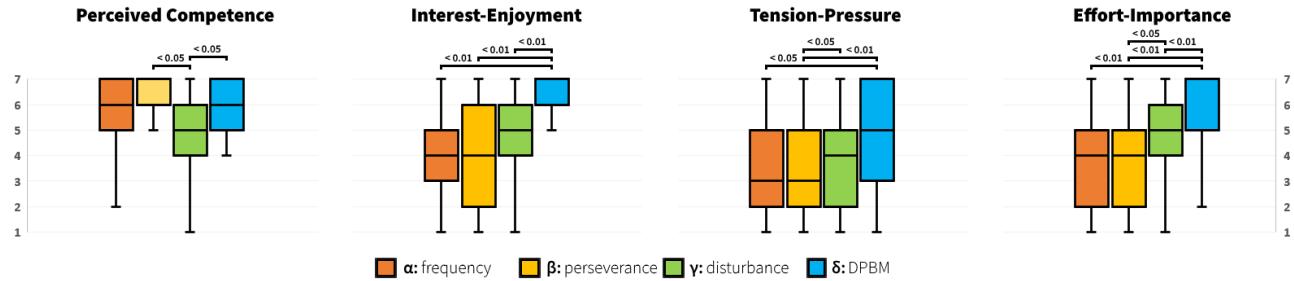
These outcomes were further evaluated using two-tailed paired t-tests (cf. Table 3, Figure 4). Employing conservative Bonferroni correction,  $p$ -values were multiplied with the amount of repeated comparisons. DPBM received significantly higher scores of *interest-enjoyment* and *effort-importance* compared to all HPT opponents, resulting in mostly large effect sizes after Cohen. It also outperformed  $\alpha$  and  $\beta$  significantly in terms of *tension-pressure* with medium effect sizes.

|                             |                   |           |                 |
|-----------------------------|-------------------|-----------|-----------------|
| <i>perceived competence</i> | $F(3, 26) = 3.59$ | $p < .05$ | $\eta^2 = 0.29$ |
| <i>interest-enjoyment</i>   | $F(3, 26) = 8.75$ | $p < .01$ | $\eta^2 = 0.5$  |
| <i>tension-pressure</i>     | $F(3, 26) = 3.37$ | $p < .05$ | $\eta^2 = 0.28$ |
| <i>effort-importance</i>    | $F(3, 26) = 5.63$ | $p < .01$ | $\eta^2 = 0.39$ |

**Table 2.** ANOVA results ( $F(df_1, df_2)$ –,  $p$ –values,  $\eta^2$  for effect size) of the *Intrinsic Motivation Inventory* between the different opponents.

On average, players spent ( $M = 7.71, SD = 2.49$ ) minutes in the instance and used up to 91 ( $M = 21.1, SD = 11.6$ ) different skills against the DPBM opponent. Model training times lasted ( $M = 2018, SD = 3692$ ) ms per session with ( $M = 8.75, SD = 3.34$ ) ms per recorded skill.

To assess the objective quality of the underlying machine learning model and render it comparable to related approaches, 80% of the data recorded until any given time of entry into the instance was used for training, whereas 20% served for a routine initial test, resulting in ( $M = 60.64, SD = 22.57$ )% prediction accuracy.



**Figure 4.** Intrinsic motivation inventory (IMI) results for the compared DDA variables. Includes medians (center marks), standard deviations (boxes), minimal and maximal values (whiskers) and significant difference markers.

Compared to all 35 available instances in the game, *Eternal Challenge* (EC) became the most popular instance by daily numbers of players over the duration of the study (cf. Figure 5), as chi-square goodness of fit tests show (cf. Table 4), assuming equal proportions. Even when omitting the first quarter to counteract a presumable novelty bias in the distinct initial spike, EC still outmatched all alternatives.

|                    | DPBM   |                      | DPBM   |
|--------------------|--|----------------------|--|
| interest-enjoyment | ( <i>M</i> = 6.17,<br><i>SD</i> = 1.11)                    | effort-importance    | ( <i>M</i> = 5.87,<br><i>SD</i> = 1.63)                    |
| $\alpha$           | <i>p</i> = .000<br>( <i>M</i> = 4,<br><i>SD</i> = 1.51)    | $\alpha$             | <i>p</i> = .006<br>( <i>M</i> = 4.04,<br><i>SD</i> = 2.03) |
| $\beta$            | <i>p</i> = .001<br>( <i>M</i> = 4.04,<br><i>SD</i> = 2.06) | $\beta$              | <i>p</i> = .000<br>( <i>M</i> = 3.91,<br><i>SD</i> = 1.83) |
| $\gamma$           | <i>p</i> = .01<br>( <i>M</i> = 4.78,<br><i>SD</i> = 1.76)  | $\gamma$             | <i>p</i> = .004<br>( <i>M</i> = 4.91,<br><i>SD</i> = 1.81) |
| tension-pressure   | DPBM<br>( <i>M</i> = 4.91,<br><i>SD</i> = 2.15)            | perceived competence | DPBM<br>( <i>M</i> = 5.83,<br><i>SD</i> = 1.07)            |
| $\alpha$           | <i>p</i> = .049<br>( <i>M</i> = 3.39,<br><i>SD</i> = 1.8)  | $\alpha$             | <i>p</i> > .05<br>( <i>M</i> = 5.65,<br><i>SD</i> = 1.67)  |
| $\beta$            | <i>p</i> = .007<br>( <i>M</i> = 3.39,<br><i>SD</i> = 1.8)  | $\beta$              | <i>p</i> > .05<br>( <i>M</i> = 5.91,<br><i>SD</i> = 1.35)  |
| $\gamma$           | <i>p</i> > .05<br>( <i>M</i> = 4.17,<br><i>SD</i> = 1.85)  | $\gamma$             | <i>p</i> > .05<br>( <i>M</i> = 4.65,<br><i>SD</i> = 1.72)  |

**Table 3.** Means, standard deviations and significant t-test results after Bonferroni correction (*p*-values and Cohen's *d* for effect size, *df* = 29) of the Intrinsic Motivation Inventory between the different opponents.

For the qualitative remarks, we used a structuring content analysis after Mayring [28] to assess the effect of challenging the DPBM opponent. Players were asked to state their general impression and opinion freely, without confounding or influencing questions. From the utilizable statements, 31.8% describe an appropriate challenge (e.g. “quite easy at first but afterwards I really was busy thinking about how I approach him”, “it’s almost as good as I am”), while 9.1% depict it as slightly too high or slightly to low. 13.6% emphasize a notable entertaining factor, whereas 4.4% declare that this encounter did not appeal to them. Although the behavior or decision-making of the DPBM opponent was never explicitly stated or explained during the study, 36.4% of players ascribed the ability to learn from previous battles and the adaptation to the player’s own behavior, combos, rotations and/or strategies to their enemy (e.g. “at first he randomly used skills that I also used, later he added my combos”, “tried to replicate my own skills and techniques”, “it was hilarious when I played against myself”).

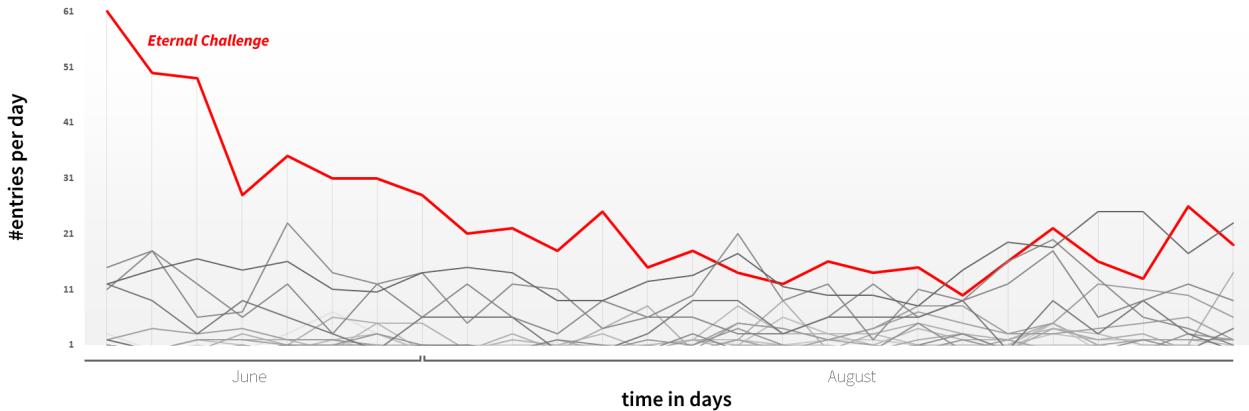
| During complete study period: |                               |                 |  |
|-------------------------------|-------------------------------|-----------------|--|
| EC vs. #2                     | $\chi^2(1, n = 1007) = 58.64$ | <i>p</i> < 0.01 |  |
| EC vs. #3                     | $\chi^2(1, n = 902) = 134.26$ | <i>p</i> < 0.01 |  |
| EC vs. #4                     | $\chi^2(1, n = 856) = 181.35$ | <i>p</i> < 0.01 |  |

| After first quarter of study period: |                              |                 |  |
|--------------------------------------|------------------------------|-----------------|--|
| EC vs. #2                            | $\chi^2(1, n = 627) = 4.48$  | <i>p</i> = 0.03 |  |
| EC vs. #3                            | $\chi^2(1, n = 526) = 45.09$ | <i>p</i> < 0.01 |  |
| EC vs. #4                            | $\chi^2(1, n = 493) = 70.93$ | <i>p</i> < 0.01 |  |

**Table 4.** Chi-square goodness of fit tests between *Eternal Challenge* and the second, third and fourth most popular instance. Less popular instances have shown similarly significant results, but have been omitted for the sake of readability.

## DISCUSSION

Our results indicate distinct effects on approval and intrinsic motivation for DPBM opponents, as well as effects on long-term commitment for the presence of DDA in general. We were successful in evidencing significantly higher motivation for players to enter adaptive instance dungeons compared to static alternatives over considerable duration of successive play sessions and report indications that DPBM attributes significantly more to this preference than traditional DDA parameter tuning.



**Figure 5.** Daily number of unique players entering *Eternal Challenge* compared to all other available instances during the study period.

This insight is based on notably high absolute IMI scores and significant differences compared to conventional DDA techniques. The outcome that DPBM outperformed HPT in terms of *interest-enjoyment* indicates a high “fun factor”, while *tension-pressure* and *effort-importance* highlight the considerable challenge, leading to an overall higher intrinsic motivation and linked potential to induce flow. The actual implicit DDA capabilities of DPBM are backed by qualitative statements that reveal an appropriate challenge, a noticeable difficulty adjustment over time and the perception of playing against an equal opponent that facilitates rethinking of habitual behavior. This work also demonstrates the technical applicability of large-scale, long-term generative player modeling with reasonable training times and accuracies.

Overall, our work provides quantitative and qualitative empirical evidence supporting our initial hypotheses about facilitating long-term motivation potential, capabilities for enabling DDA and individual representation, indicating the following responses to the respective research questions:

- We measured a substantially and consistent motivation achieved with the support of DPBM over the medium-to long-term.
- The measured intrinsic motivation of challenging a DPBM-fuelled opponent significantly exceeded traditional HPT.
- Players perceive behavior from DPBM as representative of – or comparable to – their own decision-making.

#### Limitations and Future Work

The mixed-bag fashion of the instance, which resulted from aiming to maintain an ecologically-valid realistic instance design, results in a combined experience of HPT and DPBM opponents that might influence the participants’ assertions. This study design was selected due to the long-term period of the study in a community where players know each other, rendering a between-subjects design confounding, since players would have exchanged views about the different conditions and/or complained about unjust treatments.

To further corroborate evidence and to gain a clear comparison between the different HPT factors and DPBM, the experiment should be replicated to manifest a control group (mutually exclusive from this player base) in which no DPBM (or HPT) is present. Apart from that, the Intrinsic Motivation Inventory was designed to measure single sessions within experiments. While it was not explicitly developed for this study’s setup, we found it to be the most appropriate questionnaire to assess motivation, as there is no validated reflective long-term motivation questionnaire that does not have to be raised after every single session (which was omitted in favor of ecological validity).

Based on our achievements and outcomes, we are looking forward to extending the scope of using DPBM in video games to enabling personalized, adaptive challenges that go beyond one-on-one situations to encompass interactions between different players and consider both competitive as well as cooperative interplay. DPBM agents could be deployed in multi-player scenarios where groups are challenged to deal with effects between player modeled opponents or utilized to support team-fights between human players, as equivalent reinforcements. Additionally, we plan to construct a one-dimensional proficiency metric that maps DPBM configurations to estimated competence in a game, in order to offer players more unique DDA encounters stemming from different players with similar proficiency. Using the considerably large data set recorded in this study, we seek to benchmark several alternative machine learning techniques as core mechanisms for the underlying player modeling (e.g. recurrent, deep belief, GAN or context-driven LSTM networks), to be able to give practical statements about applicability concerning temporal requirements and resulting accuracy. Furthermore, we envision DPBM as an effective instrument for elevating autonomous game testing and balancing, since actual, precise player behavior can be simulated, and temporary substitutions or continuations of disconnected players in online matches can be facilitated to minimize game experience disruptions.

## CONCLUSION

We presented the design and implementation of an adaptive instance dungeon in the MMORPG *Aion* to evaluate a novel, implicit take on *Dynamic Difficulty Adjustment* that is not dependent on manually composed parameter tuning, but affords a continually adapting challenge through *Deep Player Behavior Modeling*. In an extensive medium- to long-term study ( $n = 171$  over the course of four weeks) we contrasted an opponent applying DPBM to traditional DDA parameter tuning and can report significantly higher intrinsic motivation stemming from the unique game experience of being confronted with strategic behaviors that mirror one's own patterns. Qualitative statements reinforce the approval and positive experience of DPBM, while the consistent and dominant usage of the instance throughout the whole study period reflects its potential to elevate long-term motivation and commitment. Regarding the technical applicability of the approach, we report on the DPBM architecture, its accuracy and data structure and give an estimation about the temporal demand, yielding real-time potential.

According to the guidelines of transparent statistics, the collected data of this approach, as well as its implementation, will be made openly available upon publication, using an open-access repository.

## ACKNOWLEDGMENTS

We would like to thank all participants, testers, *Beyond Aion* for hosting the instance dungeon and Philipp Krüger for his collaboration, advice and porting. This work was funded by the German Research Foundation (DFG) as part of Collaborative Research Center (SFB) 1320 EASE - Everyday Activity Science and Engineering, University of Bremen (<http://www.easecrc.org/>), subproject H2.

## REFERENCES

- [1] Nintendo Research Development 4. 1985. *Super Mario Bros.* Game [NES]. (13 September 1985). Nintendo, Kyōto, Japan.
- [2] Ernest Adams. 2002. Balancing Games with Positive Feedback. *Gamasutra. com*, January 4 (2002).
- [3] Dennis Ang and Alex Mitchell. 2017. Comparing Effects of Dynamic Difficulty Adjustment Systems on Video Game Experience. In *Proceedings of the Annual Symposium on Computer-Human Interaction in Play*. ACM, 317–327.
- [4] Dennis Ang and Alex Mitchell. 2019. Representation and Frequency of Player Choice in Player-Oriented Dynamic Difficulty Adjustment Systems. In *Proceedings of the Annual Symposium on Computer-Human Interaction in Play*. ACM, 589–600.
- [5] ArenaNet. 2005. *Guild Wars*. Game [PC]. (28 April 2005). ArenaNet, Bellevue, WA. Played 2019.
- [6] BioWare. 2012. *Mass Effect 3*. Game [PC,XBox360,PS3,WiiU]. (6 March 2012). BioWare, Edmonton, Kanada.
- [7] Capcom Production Studio 4. 2005. *Resident Evil 4*. Game [Gamecube]. (11 January 2005).
- [8] Thomas Constant and Guillaume Levieux. 2019. Dynamic Difficulty Adjustment Impact on Players' Confidence. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, 463.
- [9] Nevermind Creations. 2019. *Korona:Nemesis*. Game [PC]. (18 August 2019).
- [10] Mihaly Csikszentmihalyi. 2013. *Flow: The psychology of happiness*. Random House.
- [11] Thomas Debeauvais. 2016. *Challenge and retention in games*. Ph.D. Dissertation. UC Irvine.
- [12] Edward L Deci, Richard Koestner, and Richard M Ryan. 1999. A meta-analytic review of experiments examining the effects of extrinsic rewards on intrinsic motivation. *Psychological bulletin* 125, 6 (1999), 627.
- [13] Anders Drachen, Alessandro Canossa, and Georgios N Yannakakis. 2009. Player modeling using self-organization in Tomb Raider: Underworld. In *2009 IEEE symposium on computational intelligence and games*. IEEE, 1–8.
- [14] Anders Drachen, Rafet Sifa, Christian Bauckhage, and Christian Thurau. 2012. Guns, swords and data: Clustering of player behavior in computer games in the wild. In *2012 IEEE conference on Computational Intelligence and Games (CIG)*. IEEE, 163–170.
- [15] Nintendo EAD. 1987. *Zelda II: The Adventure of Link*. Game [NES]. (14 January 1987). Nintendo EAD, Kyoto, Japan.
- [16] Nintendo EAD. 2014. *Mario Kart 8*. Game [WiiU,Switch]. (29 May 2014). Nintendo EAD, Kyoto, Japan. Played 2019.
- [17] William Rao Fernandes and Guillaume Levieux. 2019.  $\delta$ -logit: Dynamic Difficulty Adjustment Using Few Data Points. In *Joint International Conference on Entertainment Computing and Serious Games*. Springer, 158–171.
- [18] Julian Frommel, Fabian Fischbach, Katja Rogers, and Michael Weber. 2018. Emotion-based Dynamic Difficulty Adjustment Using Parameterized Difficulty and Self-Reports of Emotion. In *Proceedings of the 2018 Annual Symposium on Computer-Human Interaction in Play*. ACM, 163–171.
- [19] Christoffer Holmgård, Antonios Liapis, Julian Togelius, and Georgios N Yannakakis. 2014a. Evolving personas for player decision modeling. In *2014 IEEE Conference on Computational Intelligence and Games*. IEEE, 1–8.
- [20] Christoffer Holmgård, Antonios Liapis, Julian Togelius, and Georgios N Yannakakis. 2014b. Generative agents for player decision modeling in games.. In *FDG*. Citeseer.

- [21] Christoffer Holmgård, Julian Togelius, and Georgios N Yannakakis. 2013. Decision making styles as deviation from rational action: A super mario case study. In *Ninth Artificial Intelligence and Interactive Digital Entertainment Conference*.
- [22] Dayana Hristova. 2017. Dynamic difficulty adjustment (DDA) in first person shooter (FPS) games. (2017).
- [23] Robin Hunnicke. 2005. The case for dynamic difficulty adjustment in games. In *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology*. ACM, 429–433.
- [24] Robin Hunnicke and Vernell Chapman. 2004. AI for Dynamic Difficulty Adjustment in Games.
- [25] Changchun Liu, Pramila Agrawal, Nilanjan Sarkar, and Shuo Chen. 2009. Dynamic difficulty adjustment in computer games through real-time anxiety-based affective feedback. *Int. Jnl. of Human-Computer Interaction* 25, 6 (2009), 506–529.
- [26] Ricardo Lopes, Ken Hilf, Luke Jayapalan, and Rafael Bidarra. 2013. Mobile adaptive procedural content generation. In *Proceedings of the fourth workshop on Procedural Content Generation in Games (PCG 2013), Chania, Crete, Greece*.
- [27] Tobias Mahlmann, Anders Drachen, Julian Togelius, Alessandro Canossa, and Georgios N Yannakakis. 2010. Predicting player behavior in tomb raider: Underworld. In *Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games*. IEEE, 178–185.
- [28] Philipp Mayring. 2010. Qualitative inhaltsanalyse. In *Handbuch qualitative Forschung in der Psychologie*. Springer, 601–613.
- [29] Edward McAuley, Terry Duncan, and Vance V Tammen. 1989. Psychometric properties of the Intrinsic Motivation Inventory in a competitive sport setting: A confirmatory factor analysis. *Research quarterly for exercise and sport* 60, 1 (1989), 48–58.
- [30] Olana Missura and Thomas Gärtner. 2009. Player Modeling for Intelligent Difficulty Adjustment. In *Discovery Science*, João Gama, Vítor Santos Costa, Alípio Mário Jorge, and Pavel B. Brazdil (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 197–211.
- [31] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, and others. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529.
- [32] Fausto Mourato, Fernando Birra, and Manuel Próspero dos Santos. 2014. Difficulty in action based challenges: success prediction, players' strategies and profiling. In *Proceedings of the 11th Conference on Advances in Computer Entertainment Technology*. ACM, 9.
- [33] NCsoft. 2003. *Lineage 2*. Game [PC]. (1 October 2003). NCSoft, Seongnam, South Korea.
- [34] NCsoft. 2008. *Aion*. Game [PC]. (25 September 2008). NCSoft, Seongnam, South Korea. Played August 2019.
- [35] Pedro A Nogueira, Vasco Torres, Rui Rodrigues, Eugénio Oliveira, and Lennart E Nacke. 2016. Vanishing scares: biofeedback modulation of affective player experiences in a procedural horror game. *Journal on Multimodal User Interfaces* 10, 1 (2016), 31–62.
- [36] Juan Ortega, Noor Shaker, Julian Togelius, and Georgios N Yannakakis. 2013. Imitating human playing styles in super mario bros. *Entertainment Computing* 4, 2 (2013), 93–104.
- [37] Johannes Pfau, Jan David Smeddinck, Ioannis Bikas, and Rainer Malaka. 2020. Bot or not? User Perceptions of Player Substitution with Deep Player Behavior Models. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. ACM.
- [38] Johannes Pfau, Jan David Smeddinck, and Rainer Malaka. 2018. Towards Deep Player Behavior Models in MMORPGs. In *Annual Symp. on Computer-Human Interaction in Play Ext. Abstracts (CHI PLAY '18)*. ACM, New York, NY, USA, 381–92.
- [39] Johannes Pfau, Jan David Smeddinck, and Rainer Malaka. 2019. Deep Player Behavior Models: Evaluating a Novel Take on Dynamic Difficulty Adjustment. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, LBW0171.
- [40] Mike Preuss, Thomas Pfeiffer, Vanessa Volz, and Nicolas Pflanzl. 2018. Integrated Balancing of an RTS Game: Case Study and Toolbox Refinement. In *2018 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 1–8.
- [41] Andrew K Przybylski, C Scott Rigby, and Richard M Ryan. 2010. A motivational model of video game engagement. *Review of general psychology* 14, 2 (2010), 154–166.
- [42] Nintendo RD1. 2002. *Metroid Fusion*. Game [GBA]. (18 November 2002). Nintendo RD1, Kyoto, Japan.
- [43] Andrew Rollings and Ernest Adams. 2003. *Andrew Rollings and Ernest Adams on game design*. New Riders.
- [44] Robert Rosenthal and Kermit L Fode. 1963. The effect of experimenter bias on the performance of the albino rat. *Behavioral Science* 8, 3 (1963), 183–189.
- [45] Richard M Ryan and Edward L Deci. 2000. Intrinsic and extrinsic motivations: Classic definitions and new directions. *Contemporary educational psychology* 25, 1 (2000), 54–67.
- [46] Lingdao Sha, Souju He, Junping Wang, Jiajian Yang, Yuan Gao, Yidan Zhang, and Xinrui Yu. 2010. Creating appropriate challenge level game opponent by the use of dynamic difficulty adjustment. In *2010 Sixth International Conference on Natural Computation*, Vol. 8. IEEE, 3897–3901.

- [47] Noor Shaker, Julian Togelius, and Georgios N Yannakakis. 2016. The experience-driven perspective. In *Procedural Content Generation in Games*. Springer, 181–194.
- [48] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, and Laurent Sifre et al. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529 (2016), 484–489.
- [49] Adam M. Smith, Chris Lewis, Kenneth Hullet, Gillian Smith, and Anne Sullivan. 2011. An Inclusive View of Player Modeling. In *Proceedings of the 6th International Conference on Foundations of Digital Games (FDG '11)*. ACM, New York, NY, USA, 301–303.
- [50] David Stammer, Tobias Günther, and Mike Preuss. 2015. Player-adaptive spelunky level generation. In *2015 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 130–137.
- [51] Alexander Streicher and Jan D. Smeddinck. 2016. Personalized and Adaptive Serious Games. In *Entertainment Computing and Serious Games*, Ralf Dörner, Stefan Göbel, Michael Kickmeier-Rust, Maic Masuch, and Katharina Zweig (Eds.). Lecture Notes in Computer Science, Vol. 9970. Springer International Publishing, Cham, 332–377.
- [52] Mirko Suznjevic and Maja Matijasevic. 2010. Why MMORPG players do what they do: relating motivations to action categories. *International Journal of Advanced Media and Communication* 4, 4 (2010), 405–424.
- [53] Marco Tamassia, William Raffe, Rafet Sifa, Anders Drachen, Fabio Zambetta, and Michael Hitchens. 2016. Predicting player churn in destiny: A hidden markov models approach to predicting player departure in a major online game. In *2016 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 1–8.
- [54] Gerald Tesauro. 1994. TD-Gammon, a self-teaching backgammon program, achieves master-level play. *Neural computation* 6, 2 (1994), 215–219.
- [55] Valve. 2008. *Left 4 Dead*. Game [PC]. (18 November 2008). Valve, Bellevue, WA, USA. Played 2017.
- [56] Hao Wang and Chuen-Tsai Sun. 2011. Game reward systems: Gaming experiences and social meanings.. In *DiGRA Conference*, Vol. 114.
- [57] Su Xue, Meng Wu, John Kolen, Navid Aghdaie, and Kazi A Zaman. 2017. Dynamic difficulty adjustment for maximized engagement in digital games. In *Proceedings of the 26th International Conference on World Wide Web Companion*. International World Wide Web Conferences Steering Committee, 465–471.
- [58] Georgios N Yannakakis, Pieter Spronck, Daniele Loiacono, and Elisabeth André. 2013. Player modeling. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

## Review Article

# Dynamic Difficulty Adjustment (DDA) in Computer Games: A Review

Mohammad Zohaib 

*Department of Computer Science, BMS College of Engineering, Bangalore 560 019, Karnataka, India*

Correspondence should be addressed to Mohammad Zohaib; zohaib27may@gmail.com

Received 31 May 2018; Accepted 14 October 2018; Published 1 November 2018

Academic Editor: Hideyuki Nakanishi

Copyright © 2018 Mohammad Zohaib. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Dynamic difficulty adjustment (DDA) is a method of automatically modifying a game's features, behaviors, and scenarios in real-time, depending on the player's skill, so that the player, when the game is very simple, does not feel bored or frustrated, when it is very difficult. The intent of the DDA is to keep the player engrossed till the end and to provide him/her with a challenging experience. In traditional games, difficulty levels increase linearly or stepwise during the course of the game. The features such as frequency, starting levels, or rates can be set only at the beginning of the game by choosing a level of difficulty. This can, however, result in a negative experience for players as they try to map a predecided learning curve. DDA attempts to solve this problem by presenting a customized solution for the gamers. This paper provides a review of the current approaches to DDA.

## 1. Introduction

The concept of the video game is continuously changing. The early games like Computer Space and Pong of the early seventies were limited to commercial arcades, but now they are seen in multiple platforms such as cell phones, tablets, computers, and other devices. People are spending in excess of 3 billion hours weekly on gaming [1], which goes to show the extent of change it has brought to our lives.

Entertainment is but one aspect; games are now moving into reality, and the invisible boundaries separating games and reality are now becoming increasingly obscure [2]. Video games now extend to realms of healthcare [3] and education [4]. Experts have studied methods to assess how playing video games affect motor learning and its scope of improving patient involvement with therapy, especially commercial games which could be linked with specialized controls [5].

Although technology in gaming continues to evolve, a general discontent of players with the existing games has been observed due to their limitations in offering challenge levels to suit individual traits of the player like dexterity, learning and adapting ability, and emotional characteristics [6, 7]. Static levels of difficulty that are selected manually

can no longer avoid boredom in players as they, in all probability, would be unable to decide on the challenge level that matches their abilities [8]. Also, constantly calling out the players to select the difficulty levels could distract them and interrupt the game [9]. The fun factor in games depends on three factors: challenge, fantasy, and curiosity [10]. Creating an adequate level of challenge is not easy when players with varying skills are pitted against each other. When an opponent is beaten effortlessly, the game appears boring. Again, in the face of a vastly superior opponent, the game turns frustrating. These two extremes lessen fun, since an optimal challenge is not offered. Csikszentmihalyi [11] first proposed that players, when kept away from the states of boredom or frustration, travel through a "flow channel" (Figure 1) and this was incorporated into a gaming scenario by Koster [8].

This model indicates how the difficulty of a task directly relates to the performer's perception. The flow channel shows that the difficulty level can be gradually enhanced, as sufficient time exists for the players for learning and improvement to meet this challenge [11]. Thus, the model prevents the frustration of difficult situations and the boredom of simple ones. In a different study, Malone [10] suggested that if the fantasy, challenge, curiosity, and control in games could be

TABLE 1: DDA research studies since 2009.

| Year | Journal Papers | Conference Papers | Theses | Books | Total |
|------|----------------|-------------------|--------|-------|-------|
| 2009 | 1              | 2                 | 1      |       | 4     |
| 2010 | 1              | 7                 | 1      |       | 9     |
| 2011 | 2              | 5                 |        | 1     | 8     |
| 2012 | 3              | 8                 | 2      |       | 13    |
| 2013 | 2              | 5                 | 3      | 1     | 11    |
| 2014 | 1              | 4                 | 1      | 1     | 7     |
| 2015 | 1              | 5                 | 1      |       | 7     |
| 2016 | 3              | 5                 | 3      |       | 11    |
| 2017 | 5              | 7                 | 2      |       | 14    |
| 2018 | 0              | 0                 | 0      | 0     | 0     |

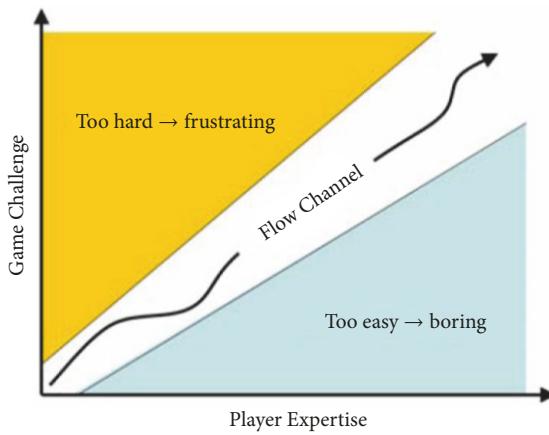


FIGURE 1: Flow channel concept proposed by Csikszentmihalyi.

balanced and associated with the gradual enhancement of difficulty level stated earlier, it could be possible that the ensuing game could keep the player entertained. Peeters [12] in her study suggested designing an automated platform for scenario-based training so that learners could engage in personalized autonomous training where agent-based notions such as beliefs, desires, and intentions can be used to deal with the gamer's competency and skills.

Numerous studies have been addressing the problems of static levels and have proposed the dynamic difficulty adjustment (DDA) technique that allows the automatic mapping of playing experience with the individual skills. DDA is a technique of automatic real-time adjustment of scenarios, parameters, and behaviors in video games, which follows the player's skill and keeps them from boredom (when the game is too easy) or frustration (when the game is too difficult). The essence of the DDA is to retain the interest of the user throughout the game and to offer a satisfactory challenge level for the player [13]. Andrade et al. suggested that DDA must cater the following three basic needs of games [14]:

- (1) The game needs to automatically track the player ability and rapidly adapt to it
- (2) The game must follow the player's improving or falling level and maintain a balance in accordance with the player's skill

(3) The process of adaptation must not be clearly perceived by the players, and successive game states need to have coherence with the earlier ones

Before applying the DDA, an understanding of the term "difficulty" is necessary. Though abstract, certain aspects need to be considered to assess and measure difficulty. Some of them are characteristics of design [15], number of resources [16], number of losses or victories [17], and so on. Nevertheless, DDA is not as easy as merely giving a player some healthier items in times of trouble. It needs an estimate of time and an entry at the right instant, as keeping the player absorbed is complicated in an interactive sense [16].

## 2. DDA Studies in the Last Ten Years

After 2009, there have been many research studies related to methods to develop or improve DDAs, including innovative applications in diverse fields. It is notable that the number of research papers in 2012 and 2017 is almost three times the number of research papers presented in 2009 (Table 1).

In this study, we have focused on DDA research studies undertaken after 2009 and have presented the important categories observed over the last decade (2009–2018). Going by the data presented in Table 1, we observe that, in the last decade, there has been a significant increase in the number of research papers on DDA over the years, and it was the highest in 2017.

Figure 2 depicts the DDA research studies carried out in the last ten years, including journal and conference papers, thesis work, and book chapters for every year.

## 3. Classification of DDA Approaches

Various methods for DDA are proposed in the literature (Table 2). The one common aspect in all methods is a requirement to measure (in a manner that may be implicit or explicit) the level of difficulty being faced by the player at any given instant. These measures are estimated by heuristic functions, also called challenge functions.

They assign a value for any game state that is indicative of the difficulty level of the game felt by the player at any given moment. Typical examples of heuristics in use are success rates of hits, numbers of pieces won and lost, life points,

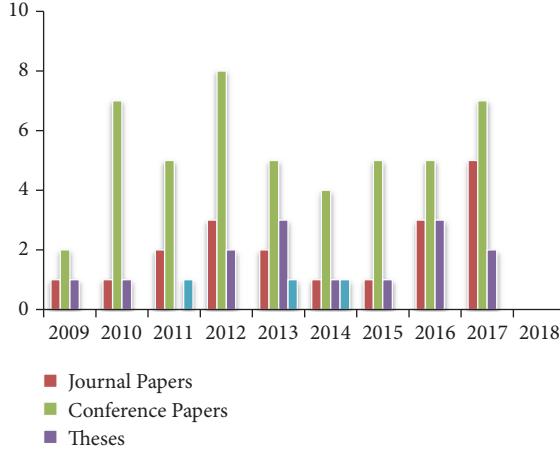


FIGURE 2: DDA studies over the past decade.

TABLE 2: List of DDA approaches.

| Author(s)                          | Approach  |
|------------------------------------|---|
| Xue et al.                         | Probabilistic Methods   |
| Pedersen, Togelius, and Yannakakis | Single and multi-layered perceptrons                            |
| Spronck et al.                     | Dynamic scripting   |
| Hunicke and Chapman                | Hamlet System   |
| Hagelback and Johansson            | Reinforcement Learning  |
| Li et al.                          | Upper Confidence Bound for Trees and Artificial Neural Networks |
| Ebrahimi and Akbarzadeh-T          | Self-organizing System and Artificial Neural Networks           |

completion time for assigned tasks, or any other metrics for calculating game scores.

There are several ways we can classify approaches to DDA.

**3.1. Probabilistic Methods.** A study on a framework that sees DDA as a problem of optimization was carried out [18]. This approach maximized player engagement all through the game. They modeled the progression of the player on a probabilistic graph (Figure 3) that maximized engagement as a well-defined objective function.

A dynamic programming technique having high efficiency was utilized to solve it. They assessed the DDA implementation using a mobile game by Electronic Arts, Inc. The group treated by DDA showed a clear increase of core engagement metrics, e.g., total number of plays and duration of game, while being revenue neutral when evaluated with the control group that did not have enabled DDA. This framework can be extended to a variety of game genres. DDA can be successfully applied to other genres if an appropriate progression model is constructed. The states for level-based games can be established by two important facets: trial and level. For games that are more complex having multiple or nonlinear progressions (e.g., role-play games), too, the states having varied dimensions can be defined. The graph would

then be more complex since more states and links would be included.

Segundo et al. [19] proposed the creation of a parameter manipulating method for DDA, which aims to enhance the pleasure of gaming. The proposed method utilizes probabilistic calculations that could be deployed in a challenge function. A sample of students was provided with a questionnaire to assess whether a significant statistical difference existed in the understanding of game difficulty, game play, and the desire to play often with and without the method. The results indicated that the DDA version showed better results than the other versions with regard to game play and the desire to play often.

In a study [20], it was proposed that both online and offline learning techniques could be used for DDA. In the offline learning, a genetic algorithm was applied to create a fuzzy rulebase for game tactics during play to manipulate the computer-controlled adversaries. In the online learning, a probabilistic method was used for adapting the game strategies to the player. The level of difficulty of the game can be adjusted in accordance with the preference of the player seeking a challenge. The results demonstrated the superior capability of the evolved offline rulebases and the efficacy of the suggested online learning method for DDA.

Bunian et al. [21] developed a modeling technique by use of data gathered from players involved in a Role-Playing Game (RPG). The proposed technique has 2 features: (i) a player's Hidden Markov Model (HMM) tracking in-game traits for modelling individual differences and (ii) use of the HMM output to generate features of behaviors for classifying real-world characteristics of players that include expertise along with the big five personality traits. The results showed the prediction capability for some of personality traits, like conscientiousness and expertise. A logistic regression model was trained considering the composition of the freshly created behavioral features for 66 participants. A three-fold cross validation was used as the dataset was small. The prediction accuracy for conscientiousness and expertise category was 59.1% and 70.13%, respectively.

Bayesian optimization techniques were used in a study [22] to design games which maximize the engagement of users. Participants were paid to attempt a game for a short period, following which they could continue to play without payment or quit voluntarily. Engagement was measured by their persistence, estimates of duration of other players, and a survey after the game. Utilizing Gaussian surrogate-based process optimization, experiments were conducted to establish game design features, especially those affecting difficulty leading to maximum engagement. The converging outcomes indicated that overt difficulty manipulations were effectual in modifying engagement only with the covert manipulations, demonstrating the user's self-perception of skill as being critical.

Hintze, Olson, and Lehman [23] proposed the idea of orthogonal coevolution and verified its effectiveness in a game that was browser-based modified from a scientific simulation. The outcomes demonstrated that evolving adversaries together with evolved friends could lead to seamless DDA and permit gamers to experience more diverse

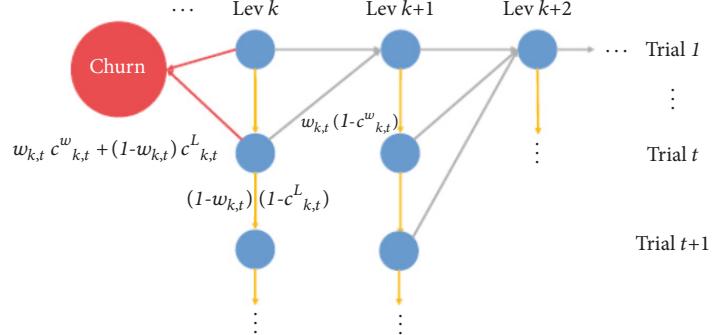


FIGURE 3: Probabilistic graph showing the player’s progression model in a typical level-based game.

situations. They concluded that such an orthogonal coevolution could be of promise for adjusting gaming difficulties.

**3.2. Single and Multilayered Perceptrons.** In a study by Pedersen, Togelius, and Yannakakis [24], the relationship between parameters of level design of platform games, player experience, and individual characteristics of play were studied. The studied design parameters had relation to the size and placement of level gaps and the presence of direction changes; and the constituents of a player’s experience comprised frustration, fun, and challenge. A neural network model, which mapped between characteristics of playing behavior, design parameters of levels, and player emotions, was trained utilizing game session data and evolutionary preference learning.

Data was gathered from the Internet. Users were inducted through messages on mailing lists and blogs and sent to a web page which contained a Java applet initiating the game and a questionnaire. After playing the games and completing the questionnaire, all the characteristics (gameplay, controllable, and player experience) were recorded in a repository on a server. After analyzing this data, they attempted a function approximation based on gameplay and controllable characteristics to record emotional choices utilizing neuroevolutionary preference learning. This data representing the function was full of noise, since the choices of the players were highly subjective and the style of playing varied. All of this, coupled with the meager training data amount, suggests the usage of a function approximator that is robust. An artificial neural network (ANN), being a nonlinear function, is a suitable option for the approximation in mapping between data and reported emotions. Therefore, a simple single-neuron (perceptron) was used to learn the relationship between characteristics (ANN data input) and the analyzed emotional choice. The primary purpose for the use of a single neuron rather than a multilayered perceptron (MLP) here was that the trained function approximator needed to be analyzed. Though MLP can approximate the function more accurately, it is simpler for us to visualize the derived function when presented by a single-neuron ANN. Learning was obtained by artificial evolution by adopting the preference learning method [25]. A generational genetic algorithm was deployed, utilizing a goodness-of-fit

function which measured the variation between the recorded emotional preferences and the corresponding model output. Results showed that there was high accuracy of prediction of challenge (77.77%), frustration (88.66%), and fun (69.18%) using a single-neuron model, which recommends using more elaborate nonlinear approximators. The study also discussed how the models generated could be used to generate game levels automatically, which would improve the player experience.

In another study, Shaker, Yannakakis, and Togelius [26] demonstrated the automatic generation of personalized levels for platform games. They built their model on the earlier work by Pedersen, Togelius, and Yannakakis [24]. At first, single layer perceptrons (SLP) were used to approximately evaluate the affective level of the players. The input subsets were selected by the sequential feature selection. To generate content customized to suit real-time player experience automatically in real-time, predicting emotions, to some extent, from controllable features is necessary. For this, the rest of the controllable features not already in the chosen feature subset were forcibly entered in the input of the multiple layer perceptron models, and the topology of the networks was made optimal for the highest accuracy of prediction.

In this study, dynamic adaptation to changes in playing styles was assessed. The model’s capacity to generalize over players of various types was tested. To carry this out, two artificial intelligence (AI) agents were deployed for play in turns, while tracking the growth of the fun value. The experiment commenced from a level generated at random. The agents played 100 levels with an agent switch after every 20 levels. The result showing the variation in fun level across 100 levels is shown in Figure 4.

It is seen that the fun value is about 70% for the initial 20 levels when the first agent plays, increases to 80% when the next agent plays for 20 levels, and drops down to 70% when the first is brought back to the game. It clearly shows the model’s capability to adjust to the player type. As a further test, the same trial was repeated on 4 human players in a reduced set of 12 levels. The result of this trial is illustrated in Figure 5, which shows the progress of fun over 48 levels. The results are similar to those obtained from the AI agents. It clearly indicates that the model robustly adapts to an individual player generalizing over various kinds of players.

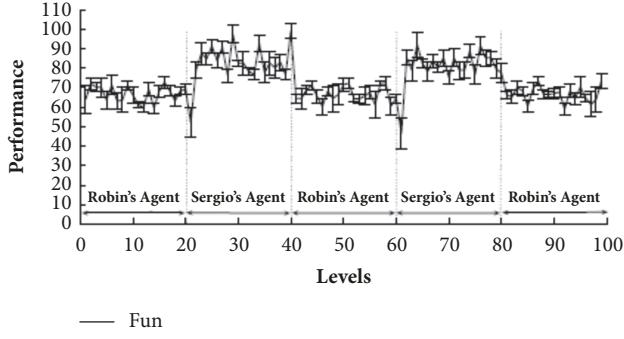


FIGURE 4: Two-agent optimized levels of fun.

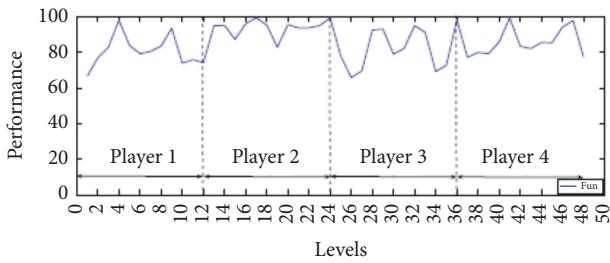


FIGURE 5: Four-player optimized levels of fun.

A study [27] constructed computational models of a player’s experience derived from interaction in gameplay for use as fitness functions for content creation in games. A classic platform game’s modified version was used for their experiments, and player data was collected from the Internet. They used the preference learning method for generating models of player experience. Feature selection was utilized to lower the features in the model. The data from training of nonlinear perceptrons was used to approximate the mapping functions between controllable features and selected gameplay. They presented the results of optimal construction of multilayer perceptrons (MLP) and the MLP model performances. They finally discussed the ways by which induced models could generate game content automatically.

Most DDA methods are based on the intuitions of designers, which do not reflect real-world playing patterns. Therefore, Jennings-Teats, Smith, and Wardrip-Fruin [28] created Polymorph that used methods from machine learning and level generation to analyze player skill and level difficulty, thereby dynamically creating levels in a 2D platformer game having continuously desired challenges. The DDA problem was addressed by generating a machine-learned difficulty model in a 2D platformer game using a model of the existing skill of the player. Multilayer Perceptrons accessed from play traces are used. These traces are gathered using a web-based tool which assigns users with various short-level components and rates them on a difficulty level. The Polymorph model utilizes the models of difficulty to choose the suitable level segment for the existing performance of the player.

Carvalho et al. [29] presented a method for generating gameplay sessions for endless games. This genre still remains largely unexplored in literature. The method uses a four-step process starting from the generation of required content to

placing the content through the gameplay sessions. A robust evaluation technique was also designed. This technique utilizes both features that can be adjusted by a designer and gameplay items gathered from gameplay sessions. The usage of 2 neural networks is a new technique that agrees with the idea of game as a service and supports it throughout the life cycle of the game. The two neural networks have different purposes: the first receives merely features that are controllable as input, and the other receives both non-controllable and controllable features as input. Both the neural networks adjust chunk (fixed-size segments of the game on which gameplay elements are placed) difficulty as their output. Thus, the first network is used in the initial development stages, where one has access to only controllable features of these chunks, and the other is used to periodically adjust the game once it is made available. Both neural networks are Multilayer Perceptrons, each having a hidden layer.

**3.3. Dynamic Scripting.** Dynamic scripting is an online unsupervised learning approach for games. It is computationally rapid, robust, efficient, and effective [30]. It operates many rulebases in the game, running one for every opponent type. These rules are designed manually utilizing domain-specific information. With the creation of a new opponent, the rules that constitute the script guiding the opponents are taken from the rulebase based on their type. The probability of a script rule selection depends on the weight value allotted to the rule. The rulebase adjusts by amending the values, reflecting the rates of failure or success of the related script rules.

In this approach, learning takes place progressively. On completing an encounter, the rule weights used in the encounter are treated depending on their effect on the result. The rules leading to success have their weights increased, while those leading to failure have their weights decreased. The remaining rules are adjusted accordingly so that the sum of all the rulebase weights remains constant. Dynamic scripting is used to generate fresh opponent tactics while increasing the level of difficulty of the game’s AI to match the level of experience of the human player (Figure 6).

There are three different enhancements to this technique allowing the opponents to learn playing a balanced game:

(1) High-fitness penalizing: The weight balancing provides rewards in proportion to the fitness value. To obtain mediocre rather than optimal behavior, the weights can be amended to reward mediocre values of fitness and punish superior values.

(2) Weight clipping: The maximum value of weight decides the highest optimization level that a learned tactic can reach. A high value for the maximum permits the weights to increase to high values, so that soon the most effective rules will nearly always be chosen. This results in scripts having near to optimal values. Similarly, low values for the maximum hamper growth of weights. This creates a large variation in scripts generated, many of which would be nonoptimal. This method automatically varies the maximum value, thereby enforcing a balanced game.

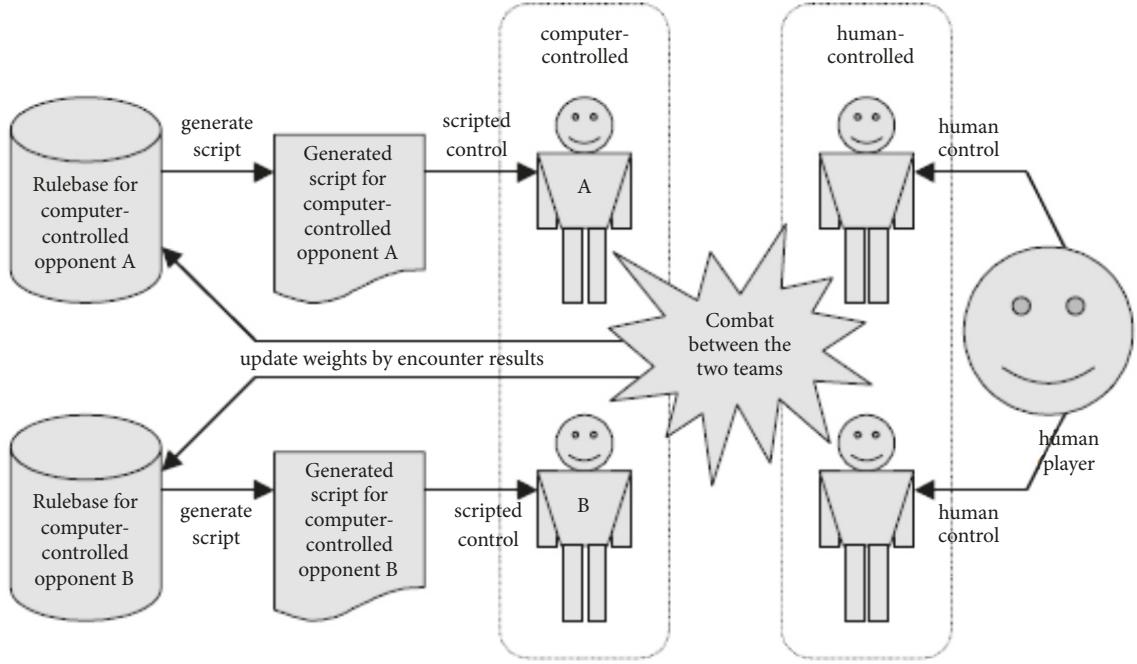


FIGURE 6: The dynamic scripting process.

(3) Top culling: Similar to weight clipping, it uses a similar mechanism for adaptation for the maximum value, the difference being that, here, weights are allowed to grow above the maximum value. However, rules having weights in excess of the maximum value do not get chosen for a generated script. As a result, frequent wins of computer-controlled opponents cause effective rules to be rejected, making opponents use weak tactics. Conversely, frequent losses would cause rules with high weights to become selectable, making opponents use weak tactics.

Experiments conducted by the authors showed that DDA by dynamic scripting is effective. It was also seen that the three approaches were tested; high-fitness penalizing was not successful, but the two other approaches did well.

**3.4. Hamlet System.** Most games use the concept of inventory, i.e., the store of items a player gathers and takes all through the game. The relative amplexness or lack of items in inventory directly impacts the experience of the players. Games are designed to control the exchange of items between the player and the world [31].

These maps of producer-consumer links can be seen as an economy or a dynamic system. Hamlet, a DDA system built by Hunicke and Chapman [13], uses methods taken from Operations Research and Inventory Management. It studies and adjusts supply and demand of the inventory in the game so as to manipulate the game difficulty. The system is essentially a group of libraries maintained in the engine of Half Life. Hamlet has functions in the following:

- (1) Managing game statistics in accordance with statistical metrics defined in advance
- (2) Deciding adjustment tasks and rules
- (3) Carrying out those tasks and rules

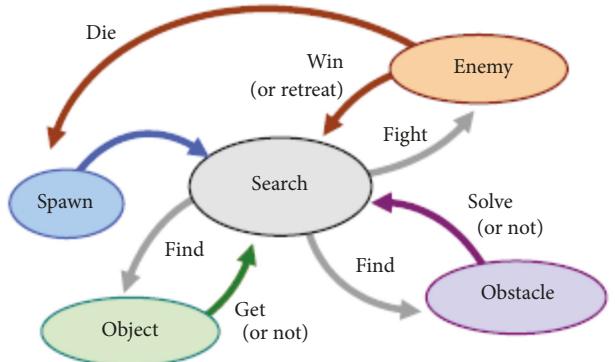


FIGURE 7: A simple FPS game state transition illustration.

- (4) Presenting data and system settings
- (5) Creating traces for playing rounds

Hamlet uses metrics for monitoring incoming game information as the players advance through the game world. It predicts the future state of the players from this information. Whenever an unwanted but preventable state is predicted, Hamlet steps in and tweaks game settings as required. Essentially, it tries to anticipate when the player is struggling repeatedly and nearing a state where his existing resources can no longer meet requirements. When this struggle is detected, Hamlet intervenes to assist the player to continue the game.

Keeping up a player's challenge and interest level is not an easy task. One popular approach is the flow model proposed by Csikszentmihalyi [11]. In an FPS environment (Figure 7), the gameplay can be illustrated with a fairly simple state transition picture.

Players engage in cycles of seeking, fetching, solving, and battle. Each new level creates new foes and hurdles. Difficulty levels and skill get enhanced with time. Hamlet is tailored to keep players within the Csikszentmihalyi's flow channel by promoting some states and demoting others. The basic aim is to keep the players in engaging loops of interaction for durations that are most appropriate based on their skill and experience gathered. To sum up, Hamlet looks to

- (1) Evaluate when adjustments are required
- (2) Decide on the changes
- (3) Make changes seamlessly

When a player is struggling, in many FPS games, it is observed that constant inventory shortfalls occur in locations where the player's existing resources do not meet the required demands. By noting trends in the inventory expenses of players, probable shortfalls are looked for, thereby identifying probable opportunities for adjustment. The evaluation process begins with the establishing of metrics to assess data. The damage data, based on its probability distribution, are analyzed. Inventory theory equations provide a basis for modeling the player's overall inventory and flow. Shortfalls are predicted based on total damage probabilities. Hamlet accordingly takes reactive and proactive action by making adjustments. Adjustment protocols are defined in the Hamlet system. Adjustment actions, together with cost estimations, form adjustment policies.

*3.5. Reinforcement Learning.* Games are played by a variety of players who use varied gaming patterns and strategies. Hence, a static game AI cannot deal with the gaming styles of all kinds of gamers. A game AI that is adaptive, therefore, can create varied gaming experiences for different playing styles and thus add interest and repeatability of play to a game. Such mechanisms have been studied with interest in recent years. For example, evolutionary algorithms by Togelius et al. [32] were used to create racing tracks that were popular with players.

Hagelback and Johansson [33] in a study observed that players enjoy playing an evenly matched game against opponents who adapt to their styles. To this end, Tan, Tan, and Tay [34] developed an adaptive AI for games that promotes even play rather than beating opponents. Here, a dynamic computer controlled opponent adapts its behavior, according to its opponent's moves. This DDA technique uses adaptive AI in the game to adjust game behaviors and parameters in real time automatically in response to the skill of the player. It can keep the player engrossed for longer periods and enhance their experience.

As mentioned, here, DDA is carried out in real time. The adaptive AI of the game requires being adept enough to make unforeseeable but rational judgments like human players but must not display the overtly thoughtless behavior. The AI also needs to be able to correctly assess its opponent in the beginning of the game itself and adjust its playing style to opponent's skill. This study proposed two adaptive algorithms, adaptive unichromosome controller (AUC) and the adaptive duochromosome controller (ADC) that utilized concepts from evolutionary computation and reinforcement learning [34] to adaptively play in real-time. Two metrics,

winning percentage difference ( $|W-L|$  and D to be minimized, where W, L, and D are wins, losses, and draws) and mean score difference ( $|s_1-s_2|$  to be minimized and  $\max(s_1, s_2)$ , where s denotes scores of players 1 and 2), were used as indicators of entertainment value. First, the game is so designed that the AI has the capability to beat the player. Second, the game AI can make deliberate mistakes, termed as artificial stupidity; therefore, the players remain interested in the game.

The training and adaptation of the AUC take place in real time when the game is in session. As its name suggests, AUC stores a single chromosome that maps to seven numbers, one corresponding to each behavior component. Each number indicates the probability of deploying a behavior component when a waypoint is crossed. The expected behavior mapped by this chromosome exemplifies a victory strategy. The chromosome tailors the proficiency of the opponent by mapping a behavior set which would be sufficient to beat him. The chromosome is initialized randomly at the beginning of every game. Whenever a waypoint is crossed, a set of rules updates the chromosome. The assumption here is that the complement of an expected victorious strategy is a losing one. The ADC and AUC are similar except that the former does not assume that the complement of an expected victorious strategy is a losing one. Instead, two sets of chromosomes are maintained, one winning and one losing chromosome, all through the game. The chromosomes are updated by different rules for wins and losses.

These controllers were tested against static controllers of varied driving traits to simulate various styles of play like heuristic controllers, neural network controllers, reverse enabled controllers, predictive fast controllers, etc. The effects of changing the mutation and learning rate were studied for both controllers (algorithms). The pattern of the difference of scores was assessed and both achieved score differences of 4 or less in at least 70.22% of the games. Wins and losses were also well scattered across the sequence of games played consecutively. It was also seen that the AUC had a low memory footprint, and the ADC was capable of maintaining a lesser number of drawn games, which could keep the player interested. The final values of the chromosomes showed that the algorithms choose various combinations of behavior components to deal with different opponents. Both controllers were able to learn proper sets of behavior components for the various opponents by way of winning percentage and mean score. Also, both were able to generalize satisfactorily to different opponents.

Sekhavat [35] suggested a personalized DDA method for a rehab game that manages difficulty settings automatically, based on a real-time patient's skills. Concepts of reinforcement learning were used as a DDA technique. It was shown that DDA has multiple objectives, in which objectives could be evaluated at different times. To solve this issue, it was proposed to use Multiple-Periodic Reinforcement Learning (MPRL) which enables the evaluation of various objectives of DDA in separate time periods. The experiments showed that MPRL performed better than available Multiple-Objective Reinforcement Learning methods in user satisfaction and enhancing the patient motor skills.

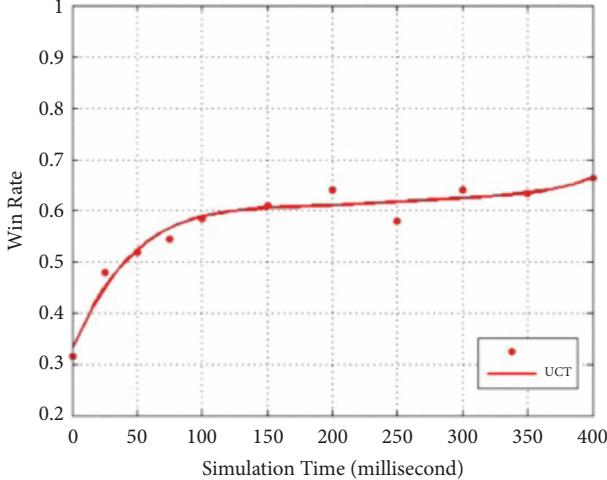


FIGURE 8: DDA from UST.

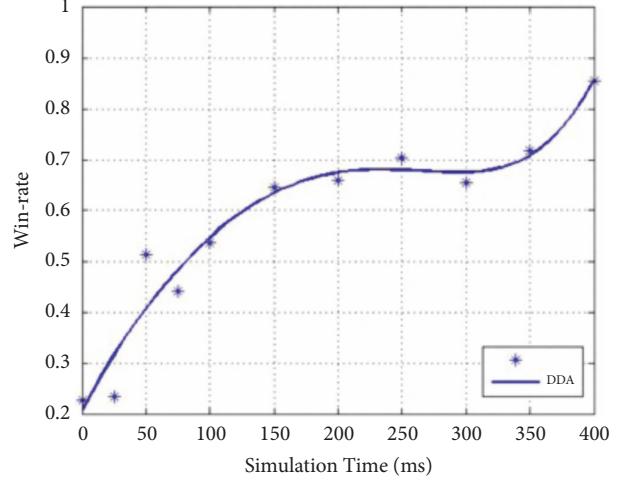


FIGURE 9: DDA from ANN.

**3.6. Upper Confidence Bound for Trees and Artificial Neural Networks.** Li et al. [36] developed a DDA technique using artificial neural networks (ANN) from data derived from the upper confidence bound for trees (UCT). The Pacman game was used as a test-bed for this study. Considering that UCT is a computing intelligence method, UCT performance significantly correlates with the duration of simulation [37]. Figure 8 illustrates DDA process from UCT-created data.

Here, the x-axis denotes simulation time, which is in the range 0–400 ms. The y-axis denotes win-rates of opponents (ghosts) which are in the range 30%–70%. The curve rises steeply in the period 0–100 ms; this period has a higher number of test data. After 100 ms, the curve flattens. The win-rate attains a maximum at 400ms. The reason for the stability of the win-rate is because of UCT being a stochastic simulation approach. In the interval 0–100 ms, the sample space is bigger and the stochastic outcomes become more accurate. Hence, the UCT performance is drastically improved. Also, after crossing 100 ms (threshold value), the accuracy of results is still fairly good so that the win-rate grows smoothly even at higher values of simulation time. UCT can be used as DDA in real-time games, too. By merely adjusting the UCT simulation time, we obtain game opponents of increasing difficulty levels.

ANN can be trained from UCT-created data. Even though the UCT approach can be deployed as DDA for games like Pacman, it is not practical to be used for complex online games because of UCT’s computational intensiveness. But then since UCT’s performance can be tweaked by varying the simulation time, ANN offline training becomes possible by running the UCT-created data with changed simulation times. Thus, DDA can be generated from UCT-created data, bypassing the computational intensiveness. In this study, the 3-Layered Feed-Forward Artificial Neural Network model in WEKA was used for the implementation.

DDA can also be created from ANN. The weights and bias of ANN are reserved in MDB files. Opponents are managed by ANN by loading MDB files.

Figure 9 illustrates the DDA from ANN. The x-axis ranging within 0–40ms is the same as Figure 7. The y-axis represents win-rate of opponents (ghosts) controlled by ANN from data created by UCT with changed simulation time ranging within 20%–86%. The curve rises steeply in the range 0–100ms. After 100ms, the curve rises steadily and the win-rate peaks at 400ms.

The performance of the opponent’s neural network depends on the training data quality. With insufficient incidences for a certain route, the ANN training remains poor. With ample incidences for all routes, the trained ANN performs well. With the increasing growth of simulation time, the UCT data achieve greater precision, which in turn creates ANN that is better trained. Comparing the two curves, we note that the DDA curve tends to rise from a minimum to a maximum simulation time. Hence, a valid DDA curve can be derived by ANN training from UCT based data. Thus, UCT is a good computation intelligence algorithm which performs better when the simulation time increases. It can therefore be used as a DDA tool by tweaking the simulation time. UCT can also create data to train ANN.

A data-driven approach for DDA was proposed by Yin et al. [38]. The objective was to match the player’s performance to the required conditions laid down by the designer. The data pertaining to dynamic game states and in-game player performance were used for taking decisions on adaptation. Trained ANNs were utilized to map the relationship between player performance, dynamic game state, adaptation decisions, and the game difficulty that resulted. The predicted difficulty enables effective adaptation of both magnitude and direction. An experiment on a training game application demonstrated the efficacy and stability of the suggested approach.

**3.7. Self-Organizing System and Artificial Neural Networks.** In another study [39], a self-organizing system (SOS) was developed, which is a group of entities that presents global system traits through local interactions while not having centralized control. This method proposes a new technique that tries to adjust the difficulty level by creating an SOS of

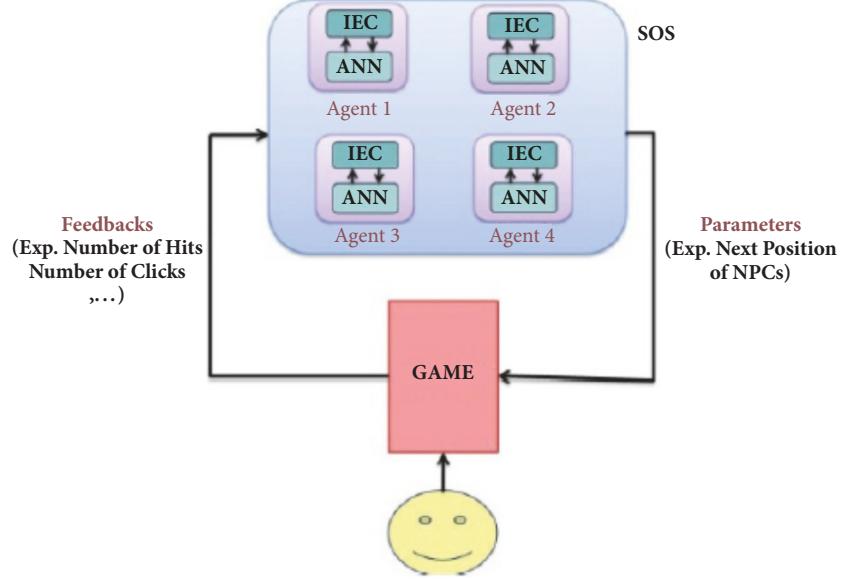


FIGURE 10: Illustration of self-organizing algorithm.

Non-Player Characters (NPCs) that are not in the player's control. To track human player traits, ANNs are used in the system. ANNs need to adapt to players having varied levels of skills and traits; therefore an evolutionary algorithm having adaptation skills was developed which modifies ANN weights (Figure 10).

Pacman game has been used as a test bed. There are two agents in the game: Pacman (the player) and ghost (opponent). The authors have considered four types of Pacmans having varying intelligence levels. The first is Cost-Based Pacman, a local agent who locates his subsequent location based on the position of his nearby ghosts. The second, named Distance-Based Pacman, is a global agent, who considers locations of all ghosts prior to deciding the next move. The third, neither fully global nor local, is named Nearest Distance-Based Pacman. The three Pacmans represent players having varying levels of skill. The fourth, Random Pacman, moves randomly and is not classified in any of these categories.

A neuroevolutionary controller for each Ghost was developed in order that they adapt to the different skill levels of the players. They decide the next position based on environmental precepts. Every ghost was given a feedforward neural network having a concealed layer. The topologies of the networks were decided during the game. Ghosts are first trained offline. This offline training helps to create chromosomes that perform better than random chromosomes. The Cooperative Coevolution Algorithm is used to train ghosts offline [40]. A subpopulation of chromosomes is considered for every ghost, which have neural network connection weights. The best performers in every subgroup are chosen as representatives. Next, to assess chromosomes of each subgroup, a game is arranged between the representatives and these chromosomes. On completion of the game, its fitness is assessed and assigned. This process is repeated till all the

chromosomes of every ghost are mapped. After assessment of all chromosomes in the subgroups is carried out, the genetic algorithm selects, crosses over, and mutates for producing new chromosomes. When the stop requirements are met, this sequence of events halts.

Next, online learning of the controllers takes place. Before the game starts, the required chromosomes are loaded in the ghost controllers. As the skill levels of the players are still unknown, intermediate chromosomes are selected for all ghosts. The subgroups are sorted on the basis of individual fitness; the median chromosomes are selected. The game begins when the chromosomes are loaded in the neural networks. The game runs for a short duration after which the system fitness is assessed. Neural controllers are trained using the Interactive Evolutionary Computation (IEC) where the fitness function replaces human evaluation [41]. As human evaluation results in fatigue, IEC optimizes systems effectively. System fitness is indirectly assessed using player feedback, e.g., number of keys pressed, occasions of key switches, and Pacman's wall hits. After each duration, these numbers are analyzed to assess fitness.

The results show that this system is capable of adapting to many skill levels by selecting proper factors that hasten convergence to the optimal requirements.

A study [42] explored the use of NEAT and rtNEAT neuroevolution techniques to create intelligent adversaries for games having real-time strategies. The primary objective is to convert the challenge created by the adversaries to match the recompetence of the player in a real-time situation, thereby resulting in a greater entertainment value experienced by the player. The study introduced the application of the neuroevolution techniques to Globulation 2 (G2), real-time strategy game for DDA. Initially, NEAT was used to optimize the functioning of G2 nonplayer characters and two suggested challenge factors were investigated by offline trials

in G2. The results indicated that the aggressiveness factors and warrior numbers are contributors to challenge because neuroevolved agents obtained succeeded in outperforming all typical AI nonplayer characters available for playing the game.

**3.8. Affective Modeling Using EEG.** Earlier researchers studied heuristic approaches based on a game state. Generally, in a game that tracks an ongoing score, decisions on DDA application can be taken when the difference between the scores of the players exceeds a threshold value, i.e., when one player becomes stronger than the other, and assuming that this would result in boredom in the stronger and frustration in the weaker player.

Stein et al. [43] have proposed a different method—measuring the excitement of players and setting in motion the game levels when the level of excitement dips below a threshold value. They have attempted to address the main issue of gaming experience directly, rather than depending on heuristic scores to decide when they are bored with the game.

An affective-state regulation technique was implemented by using headsets to decipher electroencephalography (EEG) signals and the mechanism to modify the signal to an affective state.

Next, assessing this affective state, DDA is deployed by the game. Two studies were conducted. In the first, the relationship between the EEG signals and game events (GE) was investigated. The results showed a significant correlation between the indicator for short term excitement (STE) and GE. Playing experiences were attempted to be enhanced by maximizing STE. In the second study, this EEG-initiated DDA was compared to (1) a typical heuristic technique which used elapsed duration and game status and (2) a control game without DDA. A case study was presented for the EEG-initiated DDA approached in a customized version of the Boot Camp game. The study confirmed that (1) players preferred the EEG-initiated DDA to the two other choices and (2) this method greatly increased the excitement level of the players. The study also indicated that the option of the initiating strategy is significant and greatly impacts experience of the players.

Afergan, Mikami, and Kondo [44] used functional near-infrared spectroscopy for collecting passive brain-sensing information and detecting long durations of overload or boredom. Using these physiological signals, a simulation was adapted for optimizing real-time workload that permits the system to adjust the task for the user at each moment in a better manner. To demonstrate this concept, they conducted laboratory experiments where participants, in a simulation, were assigned path planning for several unmanned aerial vehicles. The task difficulty was varied based on their state by the addition or deletion UAVs and they found that errors could be decreased by 35% over and above a baseline level. The results indicated that fNIRS brain sensing can be used to detect real-time task difficulty and an interface constructed that enhances user performance by DDA.

Fernandez et al. [45] adapted the levels of difficulty of a basic 2D platform game, working on and building levels

automatically. The method proposed consisted of DDA and Rhythm-Group Theory, a procedural content development approach, along with attention levels gathered from EEG data. Trials were planned in a manner that players needed to perform 5 varied levels automatically created by their performances and EEG data collected through a biosensor during play. Results indicated that the method adapted successfully to the difficulty levels as per the status of the player. Additionally, the method calculated difficulty utilizing calculated real-time values to decide the level.

## 4. Future Work

There are many opportunities for higher research into DDA by creation of level structures. Researchers can go beyond the standard 2D platformer video game genre and apply the same DDA concepts to different genres. Also, more research is required in new search-based techniques for identifying optimal levels. Researching player models other than agent types could be another promising area for more research. As the fitness function is a crucial element in game design, increasing its complexity by the addition of more variables that could consider many other aspects of play is yet another promising area.

An interesting research could be to investigate the possibility of covering traits like playing style. The concept of mapping the human player and developing a player model accordingly is yet another possibility. A player model that includes more behavioral aspects could yield interesting observations.

Many player modeling techniques exist currently. Integrating a few of these with present DDA approaches could open up some possibilities of interest and yield more DDA techniques tailored to a gamer's preference.

## 5. Conclusions

DDA techniques have been proven in the literature to be useful tools for incorporation in complex and dynamic systems. This investigation has presented a review on DDA applications and directions in many diverse kinds of games in the past decade highlighting some of the most representative types for every application. There are numerous application studies of DDAs in various domains, including generalizations and extensions of DDAs. The number of approaches presented here is neither complete nor exhaustive but merely a sample that demonstrates the usefulness and possible applications of AI techniques in modern video games.

## Conflicts of Interest

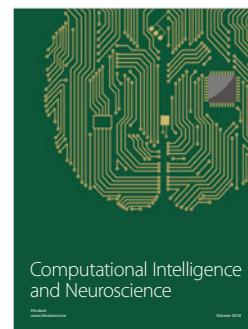
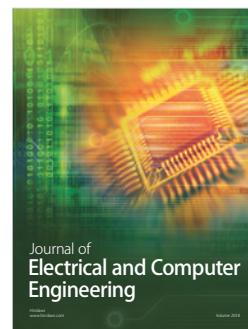
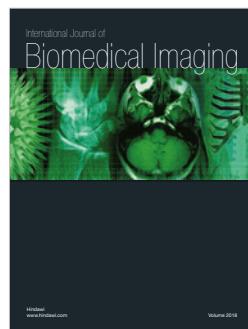
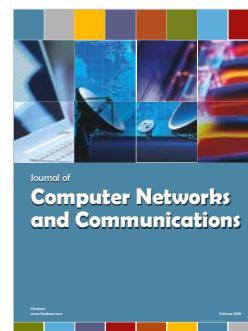
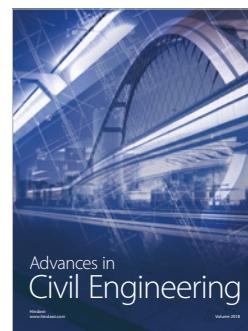
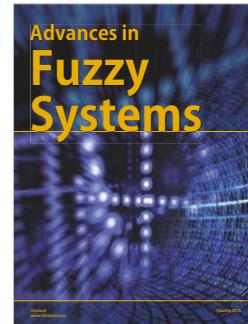
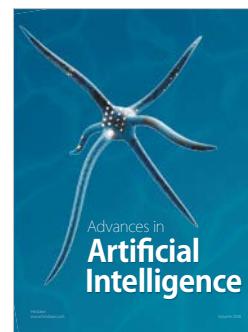
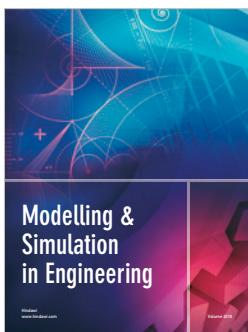
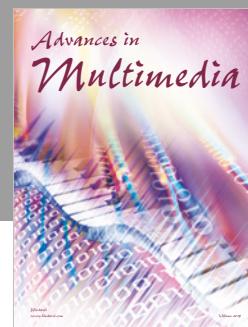
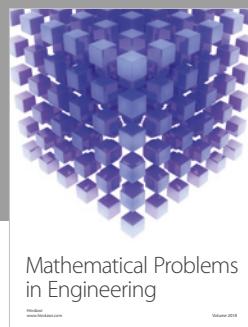
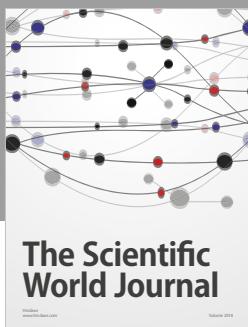
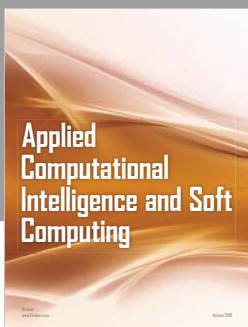
The author declares that they have no conflicts of interest.

## References

- [1] J. McGonigal, *Reality Is Broken: Why Games Make Us Better and How They Can Change The World*, Vintage, London, UK, 2012.

- [2] G. Calleja, "Digital games and escapism," *Games and Culture*, vol. 5, no. 4, pp. 335–353, 2010.
- [3] A. DeSmet, D. Thompson, T. Baranowski, A. Palmeira, M. Verloigne, and I. De Bourdeaudhuij, "Is participatory design associated with the effectiveness of serious digital games for healthy lifestyle promotion? A meta-analysis," *Journal of Medical Internet Research*, vol. 18, no. 4, 2016.
- [4] T. M. Connolly, E. A. Boyle, E. MacArthur, T. Hainey, and J. M. Boyle, "A systematic literature review of empirical evidence on computer games and serious games," *Computers & Education*, vol. 59, no. 2, pp. 661–686, 2012.
- [5] K. Lohse, N. Shirzad, A. Verster, N. Hodges, and H. F. Van der Loos, "Video Games and Rehabilitation," *Journal of Neurologic Physical Therapy*, vol. 37, no. 4, pp. 166–175, 2013.
- [6] P. Sweetser and P. Wyeth, "GameFlow," *Computers in Entertainment*, vol. 3, no. 3, 2005.
- [7] K. M. Gilledge, A. Dix, and J. Allanson, "Affective videogames and modes of affective gaming: Assist me, challenge me, emote me," in *Proceedings of the 2nd International Conference on Digital Games Research Association: Changing Views: Worlds in Play (DiGRA '05)*, 20, 16 pages, Vancouver, Canada, June 2005.
- [8] R. Koster, *A theory of fun for game design*. Sebastopol (Calif.), O'Reilly Media, 2014.
- [9] J. Chen, "Flow in games (and everything else)," *Communications of the ACM*, vol. 50, no. 4, pp. 31–34, 2007.
- [10] T. W. Malone, "Toward a theory of intrinsically motivating instruction," *Cognitive Science*, vol. 5, no. 4, pp. 333–369, 1981.
- [11] M. Csikszentmihalyi, *Flow: The Psychology of Optimal Experience*, Harper Row, New York, NY, USA, 2009.
- [12] M. M. Peeters, *Personalized Educational Games-Developing agent-supported scenario-based training [Ph.D. thesis]*, The Dutch Graduate School for Information and Knowledge Systems, 2014.
- [13] R. Hunicke and V. Chapman, "AI for Dynamic Difficulty Adjustment in Games," in *Proceedings of the Challenges in Game Artificial Intelligence AAAI Workshop*, pp. 91–96, San Jose, Calif., USA, 2004.
- [14] G. Andrade, G. Ramalho, H. Santana, and V. Corruble, "Extending reinforcement learning to provide dynamic game balancing," in *Proceedings of the Workshop on Reasoning, Representation, and Learning in Computer Games, 19th International Joint Conference on Artificial Intelligence (IJCAI)*, 12, 7 pages, Edinburgh, United Kingdom, 2005.
- [15] R. A. Bartle, *Designing Virtual Worlds*, New Riders, Berkeley, Calif., USA, 2006.
- [16] R. Hunicke, "The case for dynamic difficulty adjustment in games," in *Proceedings of the ACM SIGCHI International Conference on Advances in Computer Entertainment Technology (ACE '05)*, pp. 429–433, Valencia, Spain, June 2005.
- [17] S. Poole, *Trigger Happy Videogames and The Entertainment Revolution*, Arcade Publishing, New York, NY, USA, 2007.
- [18] S. Xue, M. Wu, J. Kolen, N. Aghdaie, and K. A. Zaman, "Dynamic Difficulty Adjustment for Maximized Engagement in Digital Games," in *Proceedings of the 26th International Conference*, pp. 465–471, Perth, Australia, April 2017.
- [19] C. V. Segundo, K. Emerson, A. Calixto, and R. P. Gusmao, "Dynamic difficulty adjustment through parameter manipulation for Space Shooter game," in *Proceedings of SB Games*, Brazil, 2016.
- [20] H. Hsieh, *Generation of Adaptive Opponents for a Predator-Prey Game*, Asia University, 2008.
- [21] S. Bunian, A. Canossa, R. Colvin, and M. S. El-Nasr, "Modeling individual differences in game behavior using HMM," in *Proceedings of the 13th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE-17)*, 2017.
- [22] M. M. Khajah, B. D. Roads, R. V. Lindsey, Y.-E. Liu, and M. C. Mozer, "Designing engaging games using Bayesian optimization," in *Proceedings of the 34th Annual Conference on Human Factors in Computing Systems, CHI 2016*, pp. 5571–5582, San Jose, Calif, USA, May 2016.
- [23] A. Hintze, R. S. Olson, and J. Lehman, "Orthogonally evolved AI to improve difficulty adjustment in video games," in *European Conference on the Applications of Evolutionary Computation*, vol. 9597 of *Lecture Notes in Computer Science*, pp. 525–540, Springer International Publishing, Cham, Switzerland, 2016.
- [24] C. Pedersen, J. Togelius, and G. N. Yannakakis, "Modeling player experience in Super Mario Bros," in *Proceedings of the 2009 IEEE Symposium on Computational Intelligence and Games (CIG)*, pp. 132–139, Milano, Italy, September 2009.
- [25] G. N. Yannakakis and J. Hallam, "Game and player feature selection for entertainment capture," in *Proceedings of the 2007 IEEE Symposium on Computational Intelligence and Games*, pp. 244–251, Honolulu, Hawaii, USA, April 2007.
- [26] N. Shaker, G. Yannakakis, and J. Togelius, "Towards automatic personalized content generation for platform games," in *Proceedings of the 6th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, AIIDE 2010*, pp. 63–68, Stanford, Calif, USA, October 2010.
- [27] C. Pedersen, J. Togelius, and G. N. Yannakakis, "Modeling player experience for content creation," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 2, no. 1, pp. 54–67, 2010.
- [28] M. Jennings-Teats, G. Smith, and N. Wardrip-Fruin, "Polymorph: A model for dynamic level generation," in *Proceedings of the 6th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, AIIDE 2010*, pp. 138–143, Stanford, Calif, USA, October 2010.
- [29] L. V. Carvalho, A. V. M. Moreira, V. V. Filho, M. Túlio, C. F. Albuquerque, and G. L. Ramalho, "A Generic Framework for Procedural Generation of Gameplay Sessions," in *Proceedings of the SB Games 2013, XII SB Games*, São Paulo, Brazil, 2013.
- [30] P. Spronck, I. Sprinkhuizen-Kuyper, and E. Postma, "Online adaptation of game opponent AI with dynamic scripting," *International Journal of Intelligent Games & Simulation*, vol. 3, no. 1, pp. 45–53, 2004.
- [31] Z. Simpson, "The In-game Economics of Ultima Online," in *Proceedings of the Game Developers Conference*, San Jose, Calif, USA, 2000.
- [32] J. Togelius, R. DeNardi, and S. M. Lucas, "Making racing fun through player modeling and track evolution," in *Proceedings of the Workshop Adaptive Approaches Optim. Player Satisfaction Comput. Phys. Games*, p. 70, 2006.
- [33] J. Hagelback and S. J. Johansson, "Measuring player experience on runtime dynamic difficulty scaling in an RTS game," in *Proceedings of the 2009 IEEE Symposium on Computational Intelligence and Games (CIG)*, pp. 46–52, Milano, Italy, September 2009.
- [34] C. H. Tan, K. C. Tan, and A. Tay, "Dynamic game difficulty scaling using adaptive behavior-based AI," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 3, no. 4, pp. 289–301, 2011.
- [35] Y. A. Sekhavat, "MPRL: Multiple-Periodic Reinforcement Learning for difficulty adjustment in rehabilitation games,"

- in *Proceedings of the 5th IEEE International Conference on Serious Games and Applications for Health, SeGAH 2017*, Perth, Australia, April 2017.
- [36] X. Li, S. He, Y. Dong et al., “To create DDA by the approach of ANN from UCT-created data,” in *Proceedings of the 2010 International Conference on Computer Application and System Modeling (ICCASM 2010)*, pp. V8-475–V8-478, Taiyuan, China, October 2010.
  - [37] J. Yang, Y. Gao, S. He et al., “To Create Intelligent Adaptive Game Opponent by Using Monte-Carlo for Tree Search,” in *Proceedings of the 2009 Fifth International Conference on Natural Computation*, pp. 603–607, Tianjian, China, August 2009.
  - [38] H. Yin, L. Luo, W. Cai, Y.-S. Ong, and J. Zhong, “A data-driven approach for online adaptation of game difficulty,” in *Proceedings of the 2015 IEEE Conference on Computational Intelligence and Games, CIG 2015*, pp. 146–153, Tainan, Taiwan, September 2015.
  - [39] A. Ebrahimi and M.-R. Akbarzadeh-T, “Dynamic difficulty adjustment in games by using an interactive self-organizing architecture,” in *Proceedings of the 2014 Iranian Conference on Intelligent Systems, ICIS 2014*, Iran, February 2014.
  - [40] M. A. Potter and K. A. de Jong, “Cooperative coevolution: an architecture for evolving coadapted subcomponents,” *Evolutionary Computation*, vol. 8, no. 1, pp. 1–29, 2000.
  - [41] H. Takagi, “Interactive evolutionary computation: fusion of the capabilities of EC optimization and human evaluation,” *Proceedings of the IEEE*, vol. 89, no. 9, pp. 1275–1296, 2001.
  - [42] J. K. Olesen, G. N. Yannakakis, and J. Hallam, “Real-time challenge balance in an RTS game using rtNEAT,” in *Proceedings of the 2008 IEEE Symposium on Computational Intelligence and Games, CIG 2008*, pp. 87–94, Perth, Australia, December 2008.
  - [43] A. Stein, Y. Yotam, R. Puzis, G. Shani, and M. Taieb-Maimon, “EEG-triggered dynamic difficulty adjustment for multiplayer games,” *Entertainment Computing*, vol. 25, pp. 14–25, 2018.
  - [44] D. Afshari, E. M. Peck, E. T. Solovey et al., “Dynamic difficulty using brain metrics of workload,” in *Proceedings of the 32nd Annual ACM Conference on Human Factors in Computing Systems, CHI 2014*, pp. 3797–3806, Toronto, Ontario, Canada, May 2014.
  - [45] H. D. B., K. Mikami, and K. Kondo, “Adaptable game experience based on player’s performance and EEG,” in *Proceedings of the 2017 Nicograph International (NicoInt)*, pp. 1–8, Kyoto, Japan, June 2017.



# Exploring Dynamic Difficulty Adjustment in Videogames

Gabriel K. Sepulveda, Felipe Besoain, and Nicolas A. Barriga

**Abstract**—Videogames are nowadays one of the biggest entertainment industries in the world. Being part of this industry means competing against lots of other companies and developers, thus, making fanbases of vital importance. They are a group of clients that constantly support your company because your video games are fun. Videogames are most entertaining when the difficulty level is a good match for the player’s skill, increasing the player engagement. However, not all players are equally proficient, so some kind of difficulty selection is required. In this paper, we will present Dynamic Difficulty Adjustment (DDA), a recently arising research topic, which aims to develop an automated difficulty selection mechanism that keeps the player engaged and properly challenged, neither bored nor overwhelmed. We will present some recent research addressing this issue, as well as an overview of how to implement it. Satisfactorily solving the DDA problem directly affects the player’s experience when playing the game, making it of high interest to any game developer, from independent ones, to 100 billion dollar businesses, because of the potential impacts in player retention and monetization.

**Keywords**—Dynamic Difficulty Adjustment, Videogames, Artificial Intelligence

## I. INTRODUCTION

WHEN someone is playing a videogame, his goal is to have fun. Game developers must infuse the games with this fun. The fun factor is composed of four axes: fantasy, curiosity, player control, and challenge. If kept in balance the player will stay entertained [1]. The challenge axis is the most difficult to control because to do so, the game difficulty and player skill must match.

Setting a single difficulty level fit for every player is not possible. A solution is to allow the player to choose a difficulty level. This method has some drawbacks, such as having a limited set of difficulty levels, creating gaps where players can fall and having difficulty progression that mismatch player learning curves. Also, by making the player aware of the change in difficulty the game experience is affected. Over the last decade, there have been multiple publications related to the improvement of this issue, using Dynamic Difficulty Adjustment (DDA) [2]. As a result, there are various algorithms the developer can use to implement DDA, and choosing the most appropriate one can be a difficult task. The lack of experience makes choosing and correctly applying the algorithm harder, leading to a poor implementation causing conflicts in the game.

G.K. Sepulveda, F. Besoain, and N.A. Barriga are with the Escuela de Ingeniería en Desarrollo de Videojuegos y Realidad Virtual, Facultad de Ingeniería, at Universidad de Talca. Campus Talca, Chile. (e-mail: gsepulveda17@alumnos.utalca.cl, fbesoain@utalca.cl, nbarriga@utalca.cl).

Corresponding Author: N.A. Barriga. (e-mail: nbarriga@utalca.cl)

In the following section we will briefly describe the Dynamic Difficulty Adjustment problem. Section III will introduce the readers to the assessment of player’s skill levels. Section IV gives an overall explanation of how a DDA system works, while section V reviews the implementations of different approaches to DDA. Finally, we close with a summary and some pointers for future work.

## II. BACKGROUND

When doing an activity that is neither boring nor frustrating, the person becomes engrossed in said activity, being able to perform longer and keep focused on the task. This state of mind is called the flow channel (flow) [3] and is present in all fields. This concept was later on introduced in the videogames area by Koster [4].

In figure 1, it’s possible to note that when the difficulty of the game is higher than the players skills the activity becomes frustrating pushing the player into a state of anxiety. In contrast when the player skills are higher than the difficulty, the game is too easy, pushing the player into a state of boredom. When neither of those happen, the user is faced by a challenge whose difficulty level matches the player’s skill, enabling him to enter the flow. Providing a series of challenges allows the player to stay in the flow for longer periods of time.

It is important to note that taking breaks between the challenges will prevent overwhelming the player. By alternating a series of constant challenges with break times it’s possible to create a game that enthralls the player and keeps him playing.

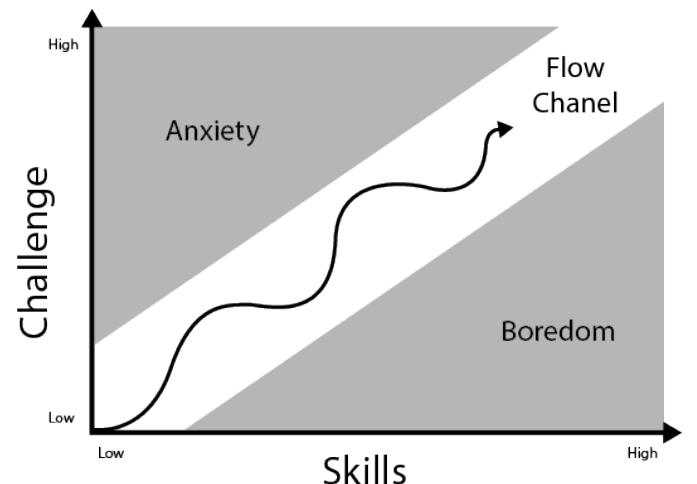


Fig. 1. Flow Channel

Hunicke proposes the creation of a system with techniques for stochastic inventory management that periodically examines the player progress and dynamically adjust the difficulty of the game challenges to adjust the player's overall experience [5]. Later on, he observes that a DDA system can improve the player experience without the need for any sophisticated AI [6].

DDA is an AI-based system that allows the change of attributes and behaviors within the game in runtime. DDA measure the player performance and change the game difficulty to match the player skills. As a result it creates the challenge used to guide the player into the flow zone. A good DDA must be able to track the player skill level and adapt to it. Changes in difficulty must follow the player learning curve and go unnoticed by the player [2].

A DDA system that fulfills these requirements can increase the player confidence in his chances to beat the game when presented with a hard challenge [7] and the core engagement of the player, resulting in an increased gameplay duration [8].

### III. ASSESSING PLAYER SKILLS

The first step needed to implement a DDA system is an evaluation of the player's performance. Through a set of predefined variables it's possible to asses if the difficulty is fit for the player. This is done by comparing the current in-game value of this variables with their expected value.

#### A. Variables

In order to choose the variables used to evaluate the player performance it is important to have a clear notion of what is considered as a failure, a success or a skill that the player needs to develop within the game. Usually, they'll correspond to game rules and win or loss conditions [9].

A good example is given in a study using Pac-Man as a test-bed. By measuring the number of hits on the maze walls, the number of keys pressed and the number of direction switches, it is possible to have an idea of what the player skills are [10]. Also the lives lost and pills collected versus time are good indicators of failure and success rate.

Another example is given by the analysis performed to the Multiplayer Online Battle Arena (MOBA) game Defense of the Ancient (DotA)<sup>1</sup> In this case the level reached versus time, and towers destroyed versus time, are used to measure success, and deaths versus time, used for failure [11]. Additionally, we can count the number of minions killed, heroes killed, items completed and missed abilities, among others, to check the player skill level.

As in the previous examples, all games can include variables that indicate the current state of the player and give information about his performance, success and failure ratio, and his learning process. The variables selected will depend on the specific game in which the DDA system is implemented.

Note that a high number of variables will result in a more precise difficulty adjustment but will also consume more memory, in the other hand a low number of variables will

result in poor adjustment. The amount of variables chosen will depend on the complexity of the game but for most of the studies a number of three to five variables is enough.

#### B. Data Collection

For the DDA system to keep track of the chosen variables, the game has to overwrite their value. The tracking can be event-triggered or permanent.

Event-triggered tracking is for variables that change when fulfilling a condition or performing an action. In this case, the method in charge of triggering the event can call the DDA system class and change the variable value.

An example of this are the variables used in the previously mentioned Pac-Man research [10]. The number of hits on maze walls, number of keys pressed or number of direction changes must be actualized just when the corresponding action happens.

On the other hand, permanent tracking is applied to variables that are always in game. The game has to call the DDA system class and update the values of the variables in the game loop. Setting a minimum time between updates is recommended to reduce processing.

Examples of variables that need permanent tracking are player health, player gold, and game progress.

#### C. Reference Point

With the collected data, it's possible to assess the player skills using the performance of a player that matches the game difficulty as a reference point. The chances of finding a player that perfectly matches the game difficulty are zero. Thus, we need to find a method that can provide the reference point.

Setting the reference point based on the beliefs of what it should be is the worst way to do so. Even with years of experience, it's unlikely to guess the correct value.

One option is to use an AI agent capable of playing the game. The data of the AI play-through will be then used as a reference point. In order to get reliable information the agent must play the game many times, a bigger number of iterations gives more precise information. An agent that plays the game perfectly isn't useful as it has to imitate a real player, to that end an agent that can play at a medium level is required instead.

However, the best option would be to have real player data. Using the same system to get data the game can be tested with real players. Using a survey after the testing session, the data of players that had fun while playing can be used as a sample.

#### D. Data Analysis

Having set the variables to analyze, the reference point for each, and being able to save the data from the player play-through, the system has the essentials needed for the evaluation of player performance. These evaluations should be made through the use of methods that return a success, failure or skill ratio. Just subtracting the number of player deaths from the expected player deaths returns a non-representative number in most cases. The evaluation must also happen constantly in

<sup>1</sup>[https://en.wikipedia.org/wiki/Defense\\_of\\_the\\_Ancients](https://en.wikipedia.org/wiki/Defense_of_the_Ancients)

the game, for the difficulty to change at the right times. To this end, the evaluation of the player performance must happen once each X number of ticks. A low X value makes the game more adaptable, but increase the process cost of the game and might not be needed, selecting the right X value will depend on the game genre.

Evaluation of player performance can be done by comparing the player's current stats with the ideal or reference ones versus a delta time [10] (equation 1).

$$\text{Difficulty} = (N - Z)/D; \quad (1)$$

$$\text{Ease} = 1 - \text{Difficulty}; \quad (2)$$

Let's take, for example, a situation where the player performs an action  $N$  times in a  $D$  period of time or ticks, with the ideal or reference value being  $Z$ . Using a value set that abides by  $0 \leq (N - Z) \leq D$  will return a normalized value, allowing an easier interpretation of the results.

As the enjoyment of a game is greater when the game is equally hard and easy, the point where these two values merge is the desired game state. A return value close to 0.5 fulfills this condition. The chances of getting the desired value are low, that's why leaving a proper margin is recommended. The margin must be defined by the game developer based on the situation. Leaving a 0.1 error margin would leave us with a range between [0.4, 0.6], where the game difficulty is acceptable.

This method also allows calculating the player global proficiency by calculating the average of all variables or a weighting of them. Consider that it is convenient for the global proficiency, or player global performance, to be a normalized value. To this end, the sum of all weights must also be normalized.

Another method is to assess the player current performance ( $C_p$ ) versus the expected player performance at that time ( $E_p[t]$ ), which would be our reference point.

$$\text{Performance} = C_p/E_p[t]; \quad (3)$$

In this manner, a value next to 1 means that the challenge is appropriate for the player. Same as in the previous method a error margin defined by the game developer is needed as getting the exact value is unlikely. For example, setting the error margin of 0.2 give us a range [0.8, 1.2]. This mean that values lower than 0.8 indicates the game is too hard and values greater than 1.2 means the game is too easy.

This method also allows to create a ranking of the player global performance by adding the results. Both methods can be modified to better suit the game developer preferences and needs.

#### IV. IMPLEMENTATION

There are several different DDA methods proposed in the literature. We will focus on the more straightforward ones, with the purpose of giving the reader some insight on how to implement the DDA system into his projects. For more information, a recent review of DDA research from 2009 to

TABLE I  
DDA APPROACHES

| Author(s)                          | Year | Approach   |
|------------------------------------|------|--|
| Hunicke and Chapman                | 2004 | Hamlet System [5]  |
| Spronck et al.                     | 2006 | Dynamic Scripting [12]                                     |
| Pedersen, Togelius, and Yannakakis | 2009 | Single and multi-layered perceptrons [13]                  |
| Hagelback and Johansson            | 2009 | Reinforcement Learning [14]<br>Upper Confidence Bound      |
| Li et al.                          | 2010 | for Trees and Artificial Neural Networks [15]              |
| Ebrahimi and Akbarzadeh-T          | 2014 | Self-organizing System and Artificial Neural Networks [10] |
| Sutoyo et al.                      | 2015 | Metrics [16]   |
| Xue et al.                         | 2017 | Probabilistic Methods [8]                                  |
| Stein et al.                       | 2018 | EEG-triggered dynamic difficulty adjustment [17]           |

2018 is available [2]. Table I summarizes the approaches found in the literature.

At the start of the game, there is no data of the player performance, and so, the developer must set the difficulty based on the average testing results. Then, the game must quickly adapt to the player performance. Playable tutorials are a good opportunity to get early information on the player performance without having to make him go through any real challenge. Afterward, the change of difficulty should happen less often and be smaller, allowing for the growth of the player skills.

The biggest challenge while changing the difficulty of the game in real time is avoiding the player noticing the change. To avoid this, performing the difficulty change at times where the player is not aware is a must. Such cases are times when the player is dead, change of scenes, features or elements the player has not yet seen. If the change is subtle enough, elements that the player is not currently seeing, non-perceptible changes as most of the element specific variable values changes, or minor behavior changes can be performed without fearing the player to notice it, as long as the changes don't happen too often, and aren't too extreme. There must be an upper and lower limit for how much a variable can change, as well as a time threshold for the changes to happen and a maximum number of changes for each update and stage. The changes to be performed can be added to a queue as functors or callbacks to be executed at a given time. Each change can have a tag that identifies the change. When a change is going to enter the queue any changes performed with the same tag will be removed, preventing two changes to affect the same element, as this is unneeded and can cause problems.

Big changes, such as the size of a room, conspicuous change of behavior with no reason in the gameplay, and others, must be executed in load screens, while changing scenes, or in sections not seen by the player.

These can also be added to the functors or callbacks queue and be executed when needed. On the other hand, changes to be performed on zones unseen to the player, but on the same

scene, can be performed immediately to prevent the player from getting to the zone with the changes undone.

The creation of mathematical functions that tells us how much the variables should be changed is probably the best way to perform these changes. The functor can receive as parameter the proficiency of the player in the corresponding field and calculate how much each variable must change to fit the player. The definition of the mathematical functions must be done with the data collected from the multiple testing phases. Creating a game with preset difficulties and make a study where players test all the difficulty levels is advisable.

Each one of the variables selected as an indicator of the player performance is directly affected by at least one factor in the game. For example, the death ratio of the player depends on the damage dealt by the enemies, the chances to evade an attack, and the aggressiveness of the enemies, among others. Linking each evaluation variable to a factor is crucial, as these factors are what is going to be changed in order to adapt the game difficulty to the player. These factors can be categorized in three sections, attributes, behaviors and events.

#### A. Attributes

Changing the value of attributes in the game is the first and most intuitive of the modifications to be done, and also, the easiest of them. Even so, the number of attributes that can create the difficulty trait in one of these variables can be immense. Specific information, like the given by reports of event-triggered parameters, can allow discerning how to properly balance the game. By comparing the player stats with the enemy who killed him, and the time of the engagement, it is possible to know if the player died because of the difference in damage, speed, range, attack ratio or others. There are also situations where the player died because his health was low before the engagement, which means that it's not the current enemy the reason for his death, but a previous one or the sum of them. The tracking of spikes in permanent game values allows a better adjustment. If there was a spike drop in health, knowing whether it was caused by a single enemy or by a group attack, can trigger the adjustment of the single enemy's stats, or the reduction in the number of enemies. If there was no spike then maybe the speed at which the waves of enemies reach the player is too fast and slowing down is required. As shown in the previous example, the traits of the characters in the game are not the only thing that can be changed, but also the number of enemies on the same zone, or even subtle things, such as the auto-aim range, the time limit for a quick time event, the dimension of a room, or the pace of the game.

#### B. Behaviors

Traits in a game can be presented not only as attributes but also as behaviors. Behaviors are the main component of complexity in a game and are not exclusive to NPCs (Non-Playable Characters), in three-in-line games the pieces have the behavior to self destruct if there are another two equal pieces at a two tiles distance (vertical or horizontal), in the same axis (x,y). Modern three-in-line have added new blocks and behaviors. These behaviors cannot be changed, as they

are the foundation of the game, but not all games have these restrictions.

In a stealth game, the watch tower can move the light following different patterns, alternating patterns, or just randomly. The more predictable the pattern is, the easier for the player. The ability to communicate with other watch towers and to detect footprints or noises are other behaviors that can be enabled, disabled, or modified to change the difficulty of the game. In a MOBA, the tower has the behavior to attack the enemies, by changing the target priority, the game changes the difficulty drastically.

#### C. Events

Events provide an alternative that doesn't require too many modifications to the existing game codebase, as does the change in attributes and behaviors, but their implementation requires more design work than the previous two, and are easier to be spotted by the player if the implementation is poor. Events are predefined occurrences that arise under certain circumstances. For example, if the player is low on health, and his performance is low, the next enemy will always drop a health potion. Conversely, if the player's health is full, and his proficiency level is high, the next enemy hit will always deal critical damage.

## V. MODELS

In this section, we will introduce a small selection of existing approaches for DDA.

#### A. Metrics

As mentioned before, it's important to identify the factors that directly influence the player performance. In the simpler use of metrics, a multiplier is applied to the variables that controls said factors. The initial value for the multipliers is to be defined by the game developer, it's recommended to use a neutral multiplicative at the start of the game as there shouldn't be any change on the factors yet. For each relationship between the variables used to evaluate player performance ( $EV$ ) and the attributes that affect the player performance ( $FV$ ), a weight is defined. Note that the maximum number of weights needed is of  $EV \times FV$ , in case that each attribute affects every variable, which means adding variables or attributes would greatly increment the number of weights to define. If possible, each variables should be affected by a few attributes, and not all of them in order to reduce the number of weights.

These weights are added to the multiplier in function of the difference between player performance ( $PP$ ) and game difficulty ( $GD$ ). The evaluation can be performed through the use of thresholds [16] or multiplying the weight with said difference  $\frac{PP}{GD}$ .

#### B. Probabilistic Methods

Probabilistic methods focus on predicting events on the game through the use of probabilistic calculations and using the probabilities in a challenge function [8].

The probabilistic calculations are used to get the expected value of factors that directly affects the player performance and act accordingly before the player faces the challenge.

As an example, in case the player is reaching a zone with 40% of his health, the probabilistic calculation will get the expected value of the total damage the player is going to suffer. This value will be returned to the challenge function that is going to evaluate whether the challenge is too difficult or too easy for the player and act accordingly before the player enters the zone.

Experiments show that the probabilistic method is effective in games organized in stages [18] or levels [8], as it allows the AI to know which calculations it has to make based on the player current location and direction or to precalculate the values of each stage.

### C. Dynamic Scripting

Dynamic Scripting is an online machine learning technique focused in the modification of behaviors of agents in the game. Dynamic Scripts (DS) are built from a set of rulebases, one for each type of agent to be modified. Each time an agent is created, the associated rulebase is used to create a new script that controls its behavior. Each rule of the rulebase has an associated weight that determine the chances of selecting the rule. The weights are adjusted based on a fitness function that evaluates the system's performance [12].

Image 2 shows an example of a Dynamic Scripting system in action. The character has an action rule-base. Variations on the same rules are performed to give the algorithm more options. The DS selects rules from the rule base to create the behavior script.

In order to have a good performance, a Dynamic Scripting implementation has to meet certain requirements [12], [19]:

**Speed:** Algorithms in online machine learning must be computationally fast since they take place during gameplay.

**Effectiveness:** The created scripts should be at least as challenging as manually designed ones.

**Robustness:** The learning mechanism must be able to cope with a significant amount of randomness inherent in most commercial gaming mechanisms.

**Efficiency:** The learning process should rely on a small number of trials, since a player experiences a limited number of encounters with similar groups of opponents.

**Clarity:** Adaptive game AI must produce easily interpretable results, because game developers distrust learning techniques of which the results are hard to understand.

**Variety:** Adaptive game AI must produce a variety of different behaviors, because agents that exhibit predictable behavior are less entertaining than agents that exhibit unpredictable behavior.

**Consistency:** The average number of learning opportunities needed for adaptive game AI to produce successful results should have a high consistency to ensure that their achievement is independent both from the behavior of the human player, and from random fluctuations in the learning process.

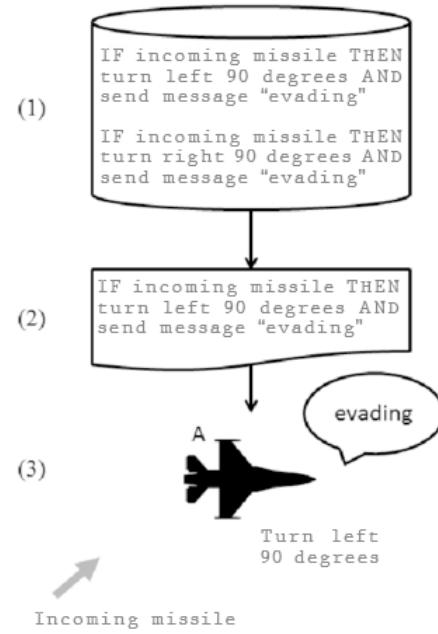


Fig. 2. Dynamic Scripting [20]

**Scalability:** Adaptive game AI must be able to scale the difficulty level of its results to the skill level of the human player.

As Dynamic Scripting is a continually learning AI, it might keep improving until it reaches a point where it can always defeat the player. In DDA the objective isn't to beat the player but to provide a fitting challenge, so in order to avoid the AI surpassing the player skills some modifications have been added to traditional Dynamic Scripting.

Weight Clipping is a technique where a maximum  $W$  value is determined, preventing the weights to grow over  $W$  [21]. Normally, a set of rules with a large win rate will result in an AI with high performance, thus increasing the weights of those rules and increasing the chances of being selected, resulting on an AI that keeps defeating the player. By setting a low  $W$  value, the chances of the rule set being selected are reduced, allowing the AI to pick tactics that make him lose against the player.

Top Culling technique, as Weight Clipping, sets a maximum  $W$  value and allows the weights to grow limitless but weights with values that surpasses  $W$  will not be selected to generate scripts [21].

Another technique used to increase the player enjoyment is the Adrenaline Rush. This technique is based on the assumption that the player's learning rate is high at the start of the game but drops through it. A learning limit is set to restrict the weights adjustment, and a maximum player fitness delta  $P$  is defined. By constantly measuring the player fitness and keeping track of the previous value a player fitness difference between the previous and current state can be calculated. When this player fitness delta drops below  $P$  the learning limit of the weights is decreased so the AI doesn't overgrow the player [22].

Finally, it's possible for the Dynamic Scripting's fitness function to minimize the difference between it's performance and the player's, rather than to maximize absolute performance. This way the AI will adjust the weights to select rules that increase the chances of the game difficulty fitting the player skills, instead of the rules that make it more likely to win.

## VI. CONCLUSION

Dynamic Difficulty Adjustment (DDA) is a technique that allows the developer to give the player a game that adapts itself to fit him, thus increasing player engagement. In recent years the amount of research in DDA has increased. As a result, there are many approaches to implement it, with the main difference being the method used to change the attributes and behaviors of the game. Because of that, it is possible to have a general structure of how to implement the DDA. The proposed structure is separated into two main parts, measuring the player proficiency and adjusting the game accordingly, using the method preferred by the developer.

Implementing a Dynamic Difficulty Adjustment system is a powerful tool that allows for a better game experience. Its implementation is based mainly in data collection from test subjects and the player playing the game, and use of metric and mathematical functions to define the changes to be performed. By adjusting variables and behaviors within the game create challenges fit to the player, but careful planning is needed. Poor implementation can result in having the opposite effect and overuse can cause high processing consumption from the game.

### A. Future Work

DDA is still a new field of research and much can be done to increase its effect in player engagement. New methods for implementation are always needed but research on optimization of already existing ones or creation of tools that enable an easier DDA implementation are also interesting.

Research can be done to reduce the process required for the evaluation of the fitness function, in order to be able to add more variables and create even more precise DDA systems. As well as the creation of a method that enable a low cost adjustment of weights allowing the developer to manipulate more factors of the game.

Integration of Behaviors Trees and Finite-State Machines to the Dynamic Scripting approach, where all possible three nodes or states and transitions are preprogrammed, and the used ones are selected by the DDA system is another promising research area.

Finally the creation of tools that allows easy DDA implementation for all game styles and that can be used in the most popular frameworks or engines for game development, such as Unity or Unreal, is a field that is still untouched.

## REFERENCES

- [1] T. W. Malone, "Toward a theory of intrinsically motivating instruction," *Cognitive Science*, vol. 5, no. 4, pp. 333–369, 1981.
- [2] Z. Mohammad, "Dynamic difficulty adjustment (DDA) in computer games: A review," *Advances in Human-Computer Interaction*, vol. 2018, no. 12, pp. 333–369, 2018.
- [3] M. Csikszentmihalyi, *Flow: The Psychology of Optimal Experience*. New York : Harper and Row, 2009.
- [4] R. Koster, *Theory of fun for game design*. O'Reilly Media, Inc., 2013.
- [5] R. Hunicke and V. Chapman, "AI for dynamic difficulty adjustment in games, challenges in game artificial intelligence aaaiworkshop," 2004.
- [6] R. Hunicke, "The case for dynamic difficulty adjustment in games," in *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology*. ACM, 2005, pp. 429–433.
- [7] T. Constant and G. Levieux, "Dynamic difficulty adjustment impact on players' confidence," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, ser. CHI '19. New York, NY, USA: ACM, 2019, pp. 463:1–463:12. [Online]. Available: <http://doi.acm.org.utalca.idm.oclc.org/10.1145/3290605.3300693>
- [8] S. Xue, M. Wu, J. Kolen, N. Aghdaie, and K. A. Zaman, "Dynamic difficulty adjustment for maximized engagement in digital games," in *Proceedings of the 26th International Conference on World Wide Web Companion*. International World Wide Web Conferences Steering Committee, 2017, pp. 465–471.
- [9] M. Hendrix, T. Bellamy-Wood, S. McKay, V. Bloom, and I. Dunwell, "Implementing adaptive game difficulty balancing in serious games," *IEEE Transactions on Games*, pp. 1–1, 2018.
- [10] A. Ebrahimi and M. Akbarzadeh-T, "Dynamic difficulty adjustment in games by using an interactive self-organizing architecture," in *2014 Iranian Conference on Intelligent Systems (ICIS)*, Feb 2014, pp. 1–6.
- [11] M. P. Silva, V. d. N. Silva, and L. Chaimowicz, "Dynamic difficulty adjustment through an adaptive AI," in *2015 14th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*, Nov 2015, pp. 173–182.
- [12] P. Spronck, M. Ponsen, I. Sprinkhuizen-Kuyper, and E. Postma, "Adaptive game AI with dynamic scripting," *Machine Learning*, vol. 63, no. 3, pp. 217–248, 2006.
- [13] C. Pedersen, J. Togelius, and G. N. Yannakakis, "Modeling player experience in super mario bros," in *2009 IEEE Symposium on Computational Intelligence and Games*, Sep. 2009, pp. 132–139.
- [14] J. Hagelback and S. J. Johansson, "Measuring player experience on runtime dynamic difficulty scaling in an rts game," in *2009 IEEE Symposium on Computational Intelligence and Games*, Sep. 2009, pp. 46–52.
- [15] Xinyu Li, Suojia He, Yue Dong, Qing Liu, Xiao Liu, Yiwen Fu, Zhiyuan Shi, and Wan Huang, "To create DDA by the approach of ANN from UCT-created data," in *2010 International Conference on Computer Application and System Modeling (ICCASM 2010)*, vol. 8, Oct 2010, pp. V8–475–V8–478.
- [16] R. Sutoyo, D. Winata, K. Oliviani, and D. M. Supriyadi, "Dynamic difficulty adjustment in tower defence," *Procedia Computer Science*, vol. 59, pp. 435–444, 2015.
- [17] A. Stein, Y. Yotam, R. Puzis, G. Shani, and M. Taieb-Maimon, "EEG-triggered dynamic difficulty adjustment for multiplayer games," *Entertainment computing*, vol. 25, pp. 14–25, 2018.
- [18] C. Segundo, K. Emerson, A. Calixto, and R. Gusmao, "Dynamic difficulty adjustment through parameter manipulation for Space Shooter game," in *Proceedings of SB Games*, 2016.
- [19] P. Spronck, I. Sprinkhuizen-Kuyper, and E. Postma, "Online adaptation of game opponent ai in simulation and in practice," in *Proceedings of the 4th International Conference on Intelligent Games and Simulation*, 2003, pp. 93–100.
- [20] A. Toubman, J. J. Roessingh, P. Spronck, A. Plaat, and J. van den Herik, "Improving air-to-air combat behavior through transparent machine learning," in *Proceedings of the IITSEC 2014 Conference*, 2014.
- [21] P. Spronck, I. Sprinkhuizen-Kuyper, and E. Postma, "Enhancing the performance of dynamic scripting in computer games," in *International Conference on Entertainment Computing*. Springer, 2004, pp. 296–307.
- [22] J. J. P. Arulraj, "Adaptive agent generation using machine learning for dynamic difficulty adjustment," in *2010 International Conference on Computer and Communication Technology (ICCCT)*, Sep. 2010, pp. 746–751.

# AI for Game Production

Mark Owen Riedl

School of Interactive Computing  
Georgia Institute of Technology  
riedl@cc.gatech.edu

Alexander Zook

School of Interactive Computing  
Georgia Institute of Technology  
a.zook@gatech.edu

**Abstract**—A number of changes are occurring in the field of computer game development: persistent online games, digital distribution platforms and portals, social and mobile games, and the emergence of new business models have pushed game development to put heavier emphasis on the live operation of games. Artificial intelligence has long been an important part of game development practices. The forces of change in the industry present an opportunity for Game AI to have new and profound impact on game production practices. Specifically, Game AI agents should act as “producers” responsible for managing a long-running set of live games, their player communities, and real-world context. We characterize a confluence of four major forces at play in the games industry today, together producing a wealth of data that opens unique research opportunities and challenges for Game AI in game production. We enumerate 12 new research areas spawned by these forces and steps toward how they can be addressed by data-driven Game AI Producers.

## I. INTRODUCTION

Over the past several years the game industry has undergone a series of major changes. The increasing prominence of persistent online games, digital distribution platforms and portals, mobile and social games, and the emergence of new business models have all changed fundamental aspects of making and playing games. Developers increasingly focus on the live operation of a game, rather than creating a boxed and finalized product. Players are more diverse, have access to games in more places and at more times, and produce more data and content for developers to leverage than ever before. Across these changes four forces have come to the fore:

- 1) Games and cross-game play is increasingly persistent and presenting longer-term experiences
- 2) Game developers are starting to see their game titles as an ecosystem wherein players may move from game to game
- 3) Player communities and social gameplay have gained prominence
- 4) Developers and players have a growing interest in coupling the real world and virtual world(s)

Some of these forces have been around for a number of years, while others are just beginning to emerge. The consequence of these forces is a profound opportunity for Artificial Intelligence (AI), Computational Intelligence (CI), and Machine Learning (ML) in games (collectively referred to as *Game AI*) to play an even greater role in the development of computer games and the delivery of engaging real-time experiences to players.

Through these forces we see an opportunity for Game AI to address new research questions that have the potential to dramatically impact game development practices. The most

significant consequence of the shifting landscape of computer game development is the generation of massive amounts of data. Researchers and game developers can leverage this data in a new paradigm of data-driven Game AI focused on how to use these sources of data to improve game development practices and to deliver more engaging experiences. However, we are not merely advocating that researchers and game developers adopt data-driven analogues to existing Game AI practices (although that would be a worthy endeavor). Instead, we are proposing that the market forces enumerated above are forcing the industry to change how they think about game development processes. The modern development environment presents significant challenges to scalability—of both the practice of creating games and also the size and scope of games themselves—that are readily addressed by artificial intelligent, computational intelligence, and machine learning research.

Game AI was initially about supporting the interaction between player and the game itself. Recently there has been a push for procedural content generation as a means to support and augment the game developer on a game by game basis. In this paper, we present our desiderata on the future of Game AI in which intelligent systems support the entire game production pipeline from game creation to live operation, and across a number of games and game genres. This perspective of Game AI for production does not supplant or replace prior perspectives on Game AI, but presents a new lens through which to see an expanded role for Game AI in computer game production. We will enumerate 12 novel research questions that arise when considering the role of AI as Producer and discuss steps toward how these challenges may be addressed.

## II. THE SHIFTING LANDSCAPE OF GAME DEVELOPMENT

In this section, we enumerate some of the prominent new forces that are shaping the way that computer game development companies create games. Many of these forces have arisen as a confluence of advances in computing technology and market forces that result from a maturation of the computer game industry.

*1) Persistent Games:* The rise of Massively Multiplayer Online Games (MMOGs), social games, online game portals and long-term or recurring game experiences is creating long-term data on players within games. Many games have players join, play over long periods of time, and potentially rejoin at later times. Developers are increasingly pressed to develop content that provides long-term engagement with a game, rather than a closed experience with clear beginning and end.

*2) Ecosystem of Games:* Developers have a wider variety of tools to build games and lowered barriers to distributing

games to new users through digital platforms and online portals. Together this puts greater emphasis on keeping players engaged within an ecosystem of games from a single developer, rather than focusing on experiences only within one game. Developers are driven to provide new content of multiple kinds (including meta-game objectives) that engage players across games. The growing diversity of game players has led to additional emphasis on ensuring an ecosystem of games provides a diverse range of experiences.

*3) Player Communities:* Player communities have emerged as a major driving force for the success (and failure) of games. Players generate a mass of content from reviews and walkthroughs through add-ons and full game modifications. Continually engaging communities, supporting player socializing within the community, managing how players impact one another's experiences (positively and negatively), and leveraging user-generated content are growing concerns.

*4) Coupling of Real and Virtual Worlds:* Games are more widely adopting ways to connect to the real world. Sensing systems from the Kinect and Wiimote to mobile phone GPS provide more data on players' real-world environment. Output modalities from second screen experiences and mobile phone augmented reality to virtual reality are emerging as new ways to view game content. At the same time these technologies have introduced a host of challenges around how games can interface with and use this real-world context and respond to more unstructured types of information.

Each of these forces presents new opportunities to solve problems of real-world applicability to game developers. A side effect of each of these forces is the massive amounts of data being generated about game players. While it is not necessarily the case that new research problems will require data-driven AI, CI, and ML techniques, we see this data as a tool for tackling real-world game development problems.

### III. BACKGROUND: THREE ROLES OF GAME AI

In 2001, Laird and van Lent [1] put forth their seminal argument for AI in computer games as an academic pursuit. They specifically argued that the pursuit of “human-level” AI systems could use computer games as testbeds for research because games were intermediate environments between the toy domains researchers had been using and the full complexity of the real world. While this opened Game AI as an academic endeavor, the automation of various aspects of computer games has been a part of the industrial practice of creating commercial computer games since the beginning.

*Game AI* has come to refer to the set of tools—algorithms and representations—developed specifically to aid the creation and management of interactive, real-time, digital entertainment experiences. While games are played by humans, there are a number of aspects of the game playing experience that must be automated: roles that would be best performed by humans but are not practical to do so:

- Opponents and enemies that are meant to survive for only a short time before losing.
- Non-player characters in roles that are not “fun” to play such as shopkeepers, farmers, or victims.

- Companions in single-player experiences and non-player characters in support roles.
- Drama management at scale.
- Game designer for personalized experiences at scale.

As we go down this list, Game AI is charged with taking progressively more responsibility for the quality of the human player's experience in the game.

We group Game AI approaches into three broad roles that AI, CI, or ML takes in the creation and delivery of engaging entertainment experiences. Each role targets a different stakeholder: players, designers, and producers. The first role is *AI as Actor*, in which the Game AI system mediates between the player and the game to create a compelling real time experience. The most common manifestation of AI under this paradigm is controlling or managing bots and non-player characters. The second role is *AI as Designer*, in which the AI mediates between a game designer and a single player-game system. Under this paradigm, AI systems might procedurally generate game content or adapt the game to particular players to scale the game development process. Finally, we propose a third role, *AI as Producer*, in which the AI mediates between game producers and a number of systems of players, designers, and games. Figure 1 shows how the different roles relate to each other and the three primary human stakeholders.

These three roles are not distinct phases in the pursuit of Game AI, but overlapping sets of concerns and driving problems, all of which need to be pursued individually or in unison. We see AI Producers as a superset of AI Designers, encompassing a broader set of research questions. Equivalently, we see this as a shift from Game AI for game design to Game AI for game production. In industry the differences between designers and producers have blurred as technical barriers to content production have lowered and gameplay and business (e.g. monetization and marketing) are more tightly coupled.

#### A. Artificial Intelligence as Actor

Historically, the earliest uses of artificial intelligence in computer games was to mediate between users and the game. AI served the role of an artificial human opponent or playmate, enabling play without requiring other people or filling roles humans would be loathe to fill in a game. Compared to non-Game AI, the intelligence built into games places a greater emphasis on creating engaging and entertaining experiences for users, rather than maximizing a utility function such as score or win/loss rates.

For the AI as Actor role, research has focused on non-player character (NPC) path planning and decision making. Game agent decision making emphasizes the believability of characters to support the suspension of disbelief that the player is interacting with software instead of a monster, human opponent, or human companion. These problems are still being pursued by industry developers and academic researchers.

*Drama management* is another way for AI to mediate between users and the NPCs and other aspects of a game or virtual world [2], [3], [4]. A drama manager is an omniscient agent responsible for delivering an enjoyable and coherent narrative experience to players, much as a “Dungeon Master” does the same for tabletop role playing games.

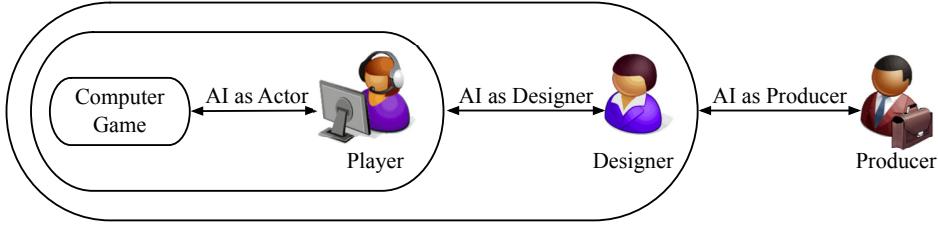


Fig. 1. Three roles of Game AI and their human stakeholders. *AI as Actor* describes how AI mediates between the player and the game itself. *AI as Designer* describes how AI mediates between a single game designer and the human-computer system of player and game. *AI as Producer* is a new role describing how AI mediates between producers and many systems of designers, users, and games.

### B. Artificial Intelligence as Designer

The second role of Game AI is to mediate between the human designer (or developer) and the human-computer system comprised of game and player. In our metaphor, game designers are responsible for building and defining a game, analyzing how players interact with the game, and iteratively refining a game to achieve a design vision. This paradigm for artificial intelligence is often referred to as *Procedural Content Generation* (PCG)—algorithms and representations for generating any and all components of games [5], [6], [7]. Offline content generation emphasizes producing content designers may curate or refine as a means of increasing the efficiency of the game development process. Online, or just-in-time, generation focuses on providing larger amounts of variation than human designers can achieve alone and/or tailoring content to a given user. A primary concern for PCG researchers has been (a) ways to appropriately represent game content to suit generation algorithms, while (b) providing means for users to interact with generation systems to author desired content and outcomes.

*Game adaptation* combines content generation and player modeling to enable AI designers to tailor games to individual players, emphasizing a closed loop of modeling player actions and automated adjustment of game content based on design goals for player behavior [8], player skill acquisition [9], or maximizing player enjoyment [10], [11], [12]. Game adaptation taken to its extreme introduces the problem of *game generation*—the automatic creation of entire games driven by (real or simulated) player feedback.

The availability of large data sets is having an impact on those who see AI as designer. In particular, *game analytics* involves capturing, aggregating, understanding, and visualizing player behavior to support designer understanding [13]. Player modeling research examines methods to describe and predict player behavior, potentially to be used by designers or automated AI systems in response [14]. While not a traditional domain of Game AI, player modeling has become increasingly prominent as AI agents shift to learning to adapt to players. Machine learning (e.g. [15], [16]) and evolutionary computing [17] are the primary areas currently being employed to perform these modeling tasks.

### C. Artificial Intelligence as Producer

The third role of Game AI uses a metaphor of AI as game producer. In our metaphor, producers concern themselves with the entire set of games and game content being made by a company, along with related aspects of managing

player communities. AI Producers mandate a shift from single player experiences within a closed game to long-term player experiences within an open game, understanding a player across multiple games in an ecosystem, and understanding how multiple players interact as an in-game and out-of-game community. AI Producers extend many methods of AI designers, driving a shift to model and adapt games that distinguishes characters (in-game avatars or personas) from players (agents manipulating those characters).

AI Producers also leverage a broader sense of context for enhancing game experiences including the community of players and the real-world context of their activities. AI for game production puts a premium on broadening the scope of Game AI to better integrate the increasingly pervasive nature of games into how games entertain and engage users. Games no longer are sharply bound by a single delivered product and Game AI ought to respond by incorporating these newly opened borders for new research domains. Effectively responding to these challenges will require new methods to leverage the masses of data being produced by players to improve interactive experiences.

## IV. GAME AI PRODUCER: RESEARCH QUESTIONS

AI producers will use the wealth of data being generated to meet the needs imposed by the four forces of (1) persistent games, (2) game ecosystems, (3) player communities, and (4) real-virtual world coupling. In each case these forces provide unique opportunities for data-driven approaches to Game AI that enhance player experience through richer sources of information and emerging modes of game-related interaction. Rather than replace earlier roles and stakeholders of Game AI, we assert that the new role of AI as Producer must emerge to address the novel concerns raised by the four forces. This new role for Game AI enhances and extends the AI techniques defined from earlier roles, leading to new research questions and techniques that potentially overlap multiple roles. In the next sections, we revisit the forces and the research questions that accompany them. Due to the overlapping nature of roles, in many cases research questions may address more than one role simultaneously.

### A. Persistent Games

Persistent games shift from closed games comprising time investments typically on the order of days to ongoing and extended game experiences spanning months or years generating long-term data on player history and activities. AI Producers address the full lifetime of a player, spanning the standard

business concerns of acquiring new players, retaining players over a long period of time, and reacquiring lapsed players. Key research questions around using long-term data to improve player engagement focus on: lifelong agents, gameplay support, and engagement-oriented content generation. Unifying these is a view of Game AI providing for long-term player engagement across an increasingly diverse group of players.

*1) How can an AI agent interact with players over very long periods of time?* An agent that persists for months, years, or even a user's lifetime is referred to as a *lifelong agent*. In the context of computer games, lifelong agents are NPCs that learn about you as a player over time. Lifelong agents serve as long term companions (or adversaries) that recognize and adapt to changes in players over time and use historical interactions with players to shift their behavior. That is, lifelong agents get to know players, and become familiar with the player, and familiar to the player.

Lifelong agents are relatively unexplored in the domain of games. In the domain of virtual agents for healthcare, Bickmore and colleagues have been designing and developing agents that interface with humans longitudinally [18]. In the context of games, lifelong agents may foster player empathy for companions—and enmity for rivals—or engage users in social interactions with game world NPCs. In addition, adapting agent behaviors to foster long-term engagement stands as a key to the problem of many online games face in creating vibrant and stable communities of players.

A central challenge for lifelong agents will be adapting to games with continually evolving content in the form of patches, expansions, downloadable content, and other incremental updates to the game the agent inhabits. Research in *lifelong machine learning* investigates how agents can transfer knowledge between particular tasks, continually learn and refine knowledge, uncover representations for complex information, and incorporate guidance or feedback from humans [19]. Addressing these challenges for game agents can provide enormous benefits to developers of live and ongoing games.

*2) How can a Game AI system support deep gameplay?* A key to persistent games is player retention. Many contemporary games have high ceilings for player skills to build up to—through complex underlying systems or ever-evolving multiplayer competition. However, complex games often lose players early in the game, especially as players increasingly have more diverse backgrounds and skills. Gameplay support agents act as mentors to players to help them overcome challenges that might otherwise cause them to quit playing a game. Gameplay support AI observes players, learns their gameplay strengths and weaknesses, and intervenes to provide players with appropriate hints, training materials, or content adjustments as needed. For example, if a player shows an inability to counter a particular strategy in an RTS the AI would identify the missing player skills and could provide instruction about the appropriate response, training video demonstrations, or set up game scenarios to practice the requisite skill. At the extreme a gameplay support agent could coach players to climb their way to the top of multiplayer competition.

Intelligent Tutoring Systems [20] present one vision for gameplay support. Applying interactive tutors to support long-term player engagement will need advances in representing

player skills, modeling players based on their game activities, devising interventions to improve those skills, and adjusting those interventions in response to tutoring success or failure. Ultimately, gameplay support systems should aspire to automatically generate tutorials—for introducing new content through coaching players along difficult missions—based on game features and gameplay data. Extracting examples appropriate to demonstrate skills from gameplay data, recognizing player skills, and choosing appropriate timing and presentation style for tutorial information are all key challenges to apply gameplay data toward automated game tutoring.

Another technique for gameplay support might lie in *Dynamic Difficulty Adjustment* (DDA). Dynamic difficulty adjustment systems make real-time adjustments to game parameters, item placement, enemy behavior, and other content to suit player abilities. Techniques for DDA have involved classical cybernetic systems [21], production systems [22], optimization of generator output from neuro-evolutionary [12] or machine learning [23] systems, or logic programming on possible player behavior [8]. Adapting these techniques for gameplay support requires capturing the long-term effects of interventions on player engagement [24], richer models of player skills related to various gameplay domains, and techniques to intelligently reuse high quality human-authored content where possible.

*3) How can a Game AI system generate motivational game content?* Content generation for long-term engagement models player values, preferences, and motivations to generate content that continues player engagement over long periods of time. The goal is to improve player retention or reacquire players who have lost interest in a game. Whereas gameplay support addresses potential “pain points” of gameplay to prevent player early dropout and improve player acquisition, long-term motivational content generation focuses on how to best retain players once they have committed to a game and encourage players who have lapsed from a game to return.

Compared to previous work on content generation, motivational content generation should incentivize players to take advantage of aspects of a game they already enjoy or to explore new elements of a game they might not have encountered. For example, generation of personalized achievements can encourage players to try alternative ways to complete a mission or level. Likewise, a system might generate mini-games within the context of a larger, persistent games based on the behaviors that a player already favors. Other forms of motivational content may exist. Regardless, a system must use longitudinal player data to determine what content to create, when to create it, and what value the content will provide to different players.

Drama managers are disembodied virtual agents that monitor virtual worlds and intervene to drive a narrative forward based on models of player experience quality [4]. Drama managers implemented in *Left 4 Dead* and *Darkspore* have demonstrated the ability to increase game replay value by varying game content and modulating player intensity levels. Drama managers that procedurally generate narratives have been demonstrated in the context of short-term experiences. Extending these systems to motivate persistent game players through narratives requires further work to personalize narratives to players through data [10], create a potentially infinite variety of narratives or quests [25], and chain narratives to

create a persistent, coherent sense of progression in an ever-growing game story.

## B. Ecosystem of Games

Digital distribution and online game portals have led to increasingly diverse games and a growing long-tail of smaller games appealing to niche interests. Ecosystems of games push entertainment goals to players' experiences across a number of potentially unrelated games, rather than within a single game. Connecting characters and gameplay from a single player across multiple games opens new opportunities in cross-game agents, cross-game content generation, and automated online game designs experiments. Previous player modeling and content generation research can play new roles when players interact with many distinct games, requiring advances in representing, collecting, and reasoning on game design knowledge.

### 4) How can AI agents interact with players across games?

Starting from the premise that lifelong virtual agents learn how to interact with individual players, we speculate that it may be advantageous for characters to interact with their players *across* many games in a company's ecosystem. To some degree, the solution involves designing for cross-game characters that manifest as recurring characters or playmates who join players across many games. However, as a lifelong agent adapts to an individual player, the question becomes one of how an agent with a constantly adapting personality, memory, and history of interactions with a player can manifest its individualized nature in the context of new games.

Cross-game agents can draw from research on competitive cross-game AI and work on socially present agents. The general game playing competition has spawned research on representing and reasoning on generic game state and rule systems to enable AI agents to play games of generic specification [26]. Cross-game agents must also consider how the personality and memory of the agent translates into new game contexts. To that end, research into *socially present agents* has explored how to control in-game behavior in reference to out-of-game context. Techniques explored include referring to historical interactions with players, simulating social roles common in a game, and explicitly signaling social and affective states not immediately relevant to in-game behavior [27]. Linking cross-game data on player-agent interactions to how social actions affect players will be crucial.

Cross-game agents may also serve as “game universe guides,” acting as a curator or tour guide to ensure players get the most out of a space of possible games. Important challenges involve understanding and eliciting player feedback on games, guiding players to new games as their interests shift, and sequencing the order of games played to optimize player experience. Cross-game data will be the linchpin to recommending different games and modeling how playing games in different orders (and ways) affects player experience.

### 5) How can a Game AI system generate cross-game content?

To date procedural content generation systems have been developed on a game-by-game basis. Cross-game content generation and adaptation explores how data acquired about a player in one game can improve content generation in another game in the ecosystem. Cross-game data enables

mining game designs for *general design knowledge*—a model mapping game mechanics to player behavior. Taken to its extreme, such design knowledge can lead to generating new games that span genres, moving beyond the current genre-focused efforts. Understanding cross-game player behavior can also allow recommending content from other games and ultimately lead to pre-adapting game experiences to players based on their behavior in other games.

### 6) How can a Game AI system add to the game ecosystem?

If AI as Producer focuses on an ecosystem of computer games, we might ask whether intelligent systems can automatically create new games that add to the ecosystem in meaningful ways. Computer game generation is a nascent area of Game AI research [28], [29], [30], [31]. Work to date has focused on a single genre at a time. Cross-game play puts a premium on new research to represent more generic game structures in a way that spans multiple genres. Many of the existing formalisms may be able to support such extensions, but how to best bridge genres remains an open question. More ambitious work will ultimately aspire to AI Producers that generate sets of games that complement one another, creating game ecosystems using a wealth of accumulated design knowledge.

Despite substantial efforts to reason on design knowledge, relatively little work has explored ways to acquire or refine general design knowledge. Current game industry efforts employ A/B testing for online game design—exemplified by Zynga’s practices. Analogous experimental methods have been used to understand how game designs impact player engagement and learning [32] or negative behavior [33]. Leveraging the unique potential of online distribution methods for *automated* rapid iteration and experimentation can open the way to addressing the challenges of extracting cross-game design knowledge. However, a number of research questions must first be addressed. First, an automated system must choose which designs to test, balancing benefits of testing against the high costs (in terms of player time, money, or negative reactions) of automated crowdsourced testing. Second, an automated system must be able to interpret the results of testing, including attributing credit/blame to aspects of design choices and appropriately changing designs in response. Third, an automated system must ultimately devise new design goals to explore when feedback indicates a given goal is unfeasible or unvaluable. Additionally, if user-generated content is available, a system might mine design principles from the content as a means of bootstrapping learning PCG systems.

## C. Community in Games

Game communities demand greater attention to how players impact one another’s experiences within a game, beyond the dynamics of cooperation and competition limited to a single session or round of play. Player communities extend in-game activities to a broader out-of-game social content. Careful *matchmaking* can group players for entertainment and engagement while *group-oriented content generation* can provide experiences tailored to sets of players. Further, communities themselves yield a wealth of user-generated content, posing opportunities to enhance interactions with PCG systems for better user- and system-generated content. However, with community comes a dark side: fraud, security violations, cheating, and abusive behavior. Game AI has an unique opportunity

to enhance negative behavior detection and automate responses beyond merely banning players.

*7) How can Game AI systems improve matchmaking and group content?* Matchmaking has traditionally focused on pairing players for competition to ensure even win rates. Online and social games, however, require grouping players for cooperative or synergistic goals. Beyond balancing win rates, players can be grouped to have complementary abilities or for purposes such as mentoring. Developing techniques to model player value as a social partner and using that information to create groups stand as key research challenges. Future developments will likely require adopting more sophisticated techniques from the social matching literature [34].

Group-oriented content generation extends PCG to the setting of engaging multiple players—with potentially conflicting interests—at once. Algorithms that balance the diverse needs of players in a given group, live game constraints (e.g. in terms of available players), and meet design goals are relevant research vectors. Both matchmaking and group-oriented content generation will require advances in modeling how players relate to one another socially and how these social and personal attributes interact with game content.

*8) How can a Game AI system reduce negative behavior?* Negative player behaviors in online games include fraud, security violations, cheating, and abusive behaviors. Fraudulent behavior commonly involves counterfeit game items sold for real-world money. Security violations compromise games through stealing players' accounts or financial data. Cheating spans hacking game systems to change their functionality to exploiting game bugs for personal gain or harm to others. Abusive behaviors include insulting other players or intentionally attempting to ruin their game experiences. Across these issues are a broad set of Game AI research challenges associated with responding appropriately to these behaviors. Note that many of these behaviors implicitly require cross-game agents—many forms of negative behavior manifest at the level of human players who act both within specific games and on game forums or other out-of-game socialization venues.

There is a wealth of research on fraud detection, security violations, and related challenges [35]. Little work, however, has addressed how Game AI agents should respond to these activities once detected. Game companies have only recently begun to systematically investigate ways to reduce negative behavior beyond banning players (e.g. [33]). We hypothesize that negative behavior can be reduced or mitigated by automatically adapting game content, rules, and mechanics to incentivize players toward pro-social behaviors. Game AI agents may minimize a player's negative impact by redirecting their actions away from others players. Detection may be improved by intentionally eliciting fraudulent or cheating behavior through an AI double-agent. Beyond negative reinforcement or punishment, Game AI agents can reward positive behavior or channel players toward more constructive pursuits.

*9) How can a Game AI system induce user-generated content?* User-generated content has become a major force for players' continuing interest in aging games. *Minecraft*, *Spore*, *Second Life*, and a host of other games have demonstrated the power of end-users to continually extend and enhance a live game. Despite the growing ubiquity of user-generated content

little research has developed methods to elicit needed content, interact with end-users to improve content, or mine design knowledge from this content.

Open questions for future user-generated content systems relate to how to best incorporate user content and feedback to improve the content created. Existing research has examined providing preference information (e.g. [36], [37]) or directly authoring the space of content (e.g. [29], [38], [39]). Enhancing user-generated content will require enabling users to teach PCG systems in new ways. *Interactive machine learning* has been developing techniques that optimize human abilities to train learning algorithms [40]. Integrating interactive machine learning approaches with user content creation interfaces can enable a new wave of learning PCG systems that improve through interaction with a player community. Key research problems include devising means to gather new modes of feedback, incorporating that feedback into PCG systems, and aggregating feedback from a diverse community of players.

#### D. Coupling the Real and Virtual Worlds

Games are increasingly coupled to the real world through input and output modalities that put a greater premium on a player's context. New input devices provide novel sensor information including GPS location (mobile devices), room layouts (Microsoft Kinect), motion data (Nintendo Wiimote, Playstation Move), sound (Kinect, webcams), and brain activity (NeuroSky, Emotiv). Output modalities have similarly become richer, including 3D displays (3D TV, Nintendo 3DS), second-screen experiences (Nintendo WiiU), virtual reality (Oculus Rift), augmented reality (mobile devices of all sorts), and potentially projection technologies. These technologies introduce nuances of player physical context including location, bodily motion data, and various physiological indicators. Beyond this information, games must also account for other aspects of player context including social circumstances or economic situation. Together this additional contextual information and output opportunities open research avenues related to incorporating real-world context into games, using out-of-game social network data in games and using games to proactively sense real-world information.

*10) How can a Game AI system utilize real-world context within a game?* Real-world data opens many avenues for game experiences that overlay on real-world settings or leverage real-world semantic information for new forms of gameplay. Human-computer interaction research has a rich literature on addressing the challenges and nuances of context, but has seen little adoption in the context of Game AI [41]. Understanding how AI agents can (or should) use context is an open problem for future Game AI agents.

Real-world settings enable new game types including augmented reality games and alternate reality games. *Augmented reality games* visually project virtual game content onto the real-world situation. *Alternate reality games* overlay fictional contexts on real-world settings without necessarily requiring a visual overlay—Google's *Ingress* is a prominent example. Both kinds of games, however, are currently circumscribed by challenges in authoring new content and fitting it to new, unanticipated real world contexts—a promising avenue for future PCG research. Preliminary work on merging the virtual

and real has explored dynamically adapting alternate reality game quests to new physical locations [42], [43]. The next stages of this work may use add other real world context in the game, such as social or economic context.

An alternative perspective on coupling real and virtual worlds involves using the semantic information present in real-world data. *Data Games* engage players in understanding semantic information in open data sets (e.g. US census data) through automatically mapping data into game content [44]. Future work should explore how in-game agents can leverage out-of-game data for richer interaction with players and tighter coupling of game worlds and real-world contexts.

**11) How can a Game AI system leverage out-of-game social networks?** Modeling in-game social interactions, social networks, and player communities can enable Game AI agents to interact with players as part of a social world that (partially) overlaps the in-game world. Current research on online game social networks has explored detecting social information including group identities [45], shared housing networks [46], and real-money trade [47]. Outside of games a wealth of prior work on modeling techniques for social and economic networks exists [48]. Here we ask: how can Game AI systems use social networks from outside a game to improve in-game experiences?

Overall, relating in-game interactions to out-of-game social interaction remains underexplored. Open research problems include uncovering how network structures can be used to draw users into a company’s game ecosystem or how to encourage users to move among games. Out-of-game social networks such as *Twitter* and *Facebook* additionally contain a wealth of user-generated material that reveals sentiments, preferences, attitudes, and non-game product usage. We speculate this data could be used to influence motivational content generation and integrate ads into game content in non-trivial ways.

**12) Can an intelligent system use games to “proactively sense” the real world?** In the age of big data, companies routinely collect data about users to improving products or otherwise monetizing user behavior. Yet, many aspects of player motivations or real-world context remain hidden from these efforts. *Proactive sensing* is the use of game content to elicit data about the real-world that might not otherwise be collected by passive observation of users. For example, consider attempting to learn whether a new coffee shop has good coffee. Game content—possibly a quest in an alternate reality game—can be generated that requires players within proximity of the coffee shop to visit and rate the shop, thus generating new data that might not otherwise have been generated. Additional applications of the proactive sensing paradigm can target other hidden real-world information such as player preferences, skills, and attitudes or semantic information about real-world objects and locations.

Proactive sensing research will require modeling the quality of information known about the world player abilities to provide that information, and challenges in generating game-relevant content and agent behaviors to elicit this information. Human computation research has modeled human abilities to perform tasks that provide computers with information about the real world [49]. Extending these approaches to games will involve incorporating player motivation to perform or

complete game tasks. Key research problems will include balancing between game design goals and information needs and generating game tasks and agent behaviors appropriate to a wide variety of information needs.

## V. CONCLUSIONS

The digital game industry is experiencing a shift to persistent games, business models emphasizing game ecosystems, more support for game communities, and new considerations for incorporating real world context into game worlds. We see these advances as positive signs of growth and maturity in the game industry. We also see these advances pushing the bounds on the scalability of game development practices.

Artificial intelligence, computational intelligence, and machine learning have always excelled at addressing problems of scalability by automating tasks and dynamically adapting system behavior. Game AI has always been an integral part of computer game development. AI Actors have enhanced player experiences by supporting players’ suspension of disbelief and dynamically managing dramatic contexts. AI Designers have supported and augmented the development of individual games through procedural content generation. We envision AI Producers taking on a new role of augmenting and scaling the game production pipeline, supporting the entire span of live operations in games, enhancing cross-game interoperations, nurturing strong player communities, and coupling real and virtual contexts. This vision is bolstered by the massive amounts of data being generated and collected by the game development industry.

The twelve new Game AI research questions here require game developers to see Game AI as part of live game production. AI for game production does not supplant previous challenges in Game AI—it extends the scope of the field of Game AI as a whole. Addressing these problems with credible AI solutions will result in immediate relevance to game developers and may present a new vector for industry game development and academic Game AI research collaboration.

## REFERENCES

- [1] J. Laird and M. VanLent, “Human-level AI’s killer application: Interactive computer games,” *AI magazine*, vol. 22, no. 2, p. 15, 2001.
- [2] M. Mateas, “An Oz-centric review of interactive drama and believable agents,” in *Artificial Intelligence Today*, ser. Lecture Notes in Computer Science, M. J. Wooldridge and M. Veloso, Eds. Springer Berlin Heidelberg, 1999, vol. 1600, pp. 297–328.
- [3] D. L. Roberts and C. L. Isbell, “A survey and qualitative analysis of recent advances in drama management,” *International Transactions on Systems Science and Applications, Special Issue on Agent Based Systems for Human Learning*, vol. 4, no. 2, pp. 61–75, 2008.
- [4] M. O. Riedl and V. Bulitko, “Interactive narrative: An intelligent systems approach,” *AI Magazine*, 2013.
- [5] M. Hendrikx, S. Meijer, J. Van Der Velden, and A. Iosup, “Procedural content generation for games: A survey,” *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 9, no. 1, p. 1, 2013.
- [6] J. Togelius, G. Yannakakis, K. Stanley, and C. Browne, “Search-based procedural content generation: A taxonomy and survey,” *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 3, no. 3, pp. 172–186, 2011.
- [7] G. N. Yannakakis, “Game AI revisited,” in *Computing Frontiers*, 2012, pp. 285–292.

- [8] A. M. Smith, E. Andersen, M. Mateas, and Z. Popović, “A case study of expressively constrainable level design automation tools for a puzzle game,” in *7th International Conference on the Foundations of Digital Games*, 2012, pp. 156–163.
- [9] E. Andersen, S. Gulwani, and Z. Popovic, “A trace-based framework for analyzing and synthesizing educational progressions,” in *ACM SIGCHI Conference on Human Factors in Computing Systems*, 2013.
- [10] H. Yu and M. O. Riedl, “A sequential recommendation approach for interactive personalized story generation,” in *11th International Conference on Autonomous Agents and Multiagent Systems*, 2012, pp. 71–78.
- [11] D. Thue, V. Bulitko, M. Spetch, and E. Wasylshen, “Interactive storytelling: A player modelling approach,” in *3rd AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2007.
- [12] N. Shaker, G. N. Yannakakis, and J. Togelius, “Towards automatic personalized content generation for platform games,” in *6th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2010.
- [13] M. Seif El-Nasr, A. Drachen, and A. Canossa, Eds., *Game Analytics*. Springer London, 2013.
- [14] A. Smith, C. Lewis, K. Hullett, G. Smith, and A. Sullivan, “An inclusive view of player modeling,” in *6th International Conference on the Foundations of Digital Games*, 2011.
- [15] B. Harrison and D. L. Roberts, “Using sequential observations to model and predict player behavior,” in *6th International Conference on the Foundations of Digital Games*, 2011.
- [16] B. G. Weber, M. Mateas, and A. H. Jhala, “Using data mining to model player experience,” in *FDG Workshop on Evaluating Player Experience*, 2011.
- [17] G. Yannakakis and J. Togelius, “Experience-driven procedural content generation,” *IEEE Transactions on Affective Computing*, vol. 2, no. 3, pp. 147–161, 2011.
- [18] T. Bickmore and D. Schulman, “A virtual laboratory for studying long-term relationships between humans and virtual agents,” in *8th International Conference on Autonomous Agents and Multiagent Systems*, 2009, pp. 297–304.
- [19] D. L. Silver, Q. Yang, and L. Li, “Lifelong machine learning systems: Beyond learning algorithms,” in *AAAI Spring Symposium Series*, 2013.
- [20] K. VanLehn, “The behavior of tutoring systems,” *International Journal of Artificial Intelligence in Education*, vol. 16, no. 3, pp. 227–265, 2006.
- [21] R. Hunicke and V. Chapman, “AI for dynamic difficulty adjustment in games,” in *AAAI Workshop on Challenges in Game Artificial Intelligence*, 2004.
- [22] B. Magerko, B. Stensrud, and L. Holt, “Bringing the schoolhouse inside the box - a tool for engaging, individualized training,” in *25th Army Science Conference*, 2006.
- [23] H. Yu and T. Trawick, “Personalized procedural content generation to minimize frustration and boredom based on ranking algorithm,” in *7th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2011.
- [24] A. Zook and M. O. Riedl, “A temporal data-driven player model for dynamic difficulty adjustment,” in *8th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2012.
- [25] B. Li, S. Lee-Urban, D. S. Appling, and M. O. Riedl, “Crowdsourcing narrative intelligence,” *Advances in Cognitive Systems*, vol. 2, pp. 25–42, 2012.
- [26] M. Genesereth, N. Love, and B. Pell, “General game playing: Overview of the AAAI competition,” *AI magazine*, vol. 26, no. 2, p. 62, 2005.
- [27] A. Pereira, R. Prada, and A. Paiva, “Socially present board game opponents,” in *Advances in Computer Entertainment*. Springer, 2012, pp. 101–116.
- [28] J. Togelius and J. Schmidhuber, “An experiment in automatic game design,” in *IEEE Symposium on Computational Intelligence and Games*, 2008, pp. 111–118.
- [29] A. M. Smith and M. Mateas, “Variations Forever: Flexibly generating rulesets from a sculptable design space of mini-games,” in *IEEE Conference on Computational Intelligence and Games*, 2010.
- [30] M. Cook, S. Colton, and J. Gow, “Initial results from co-operative co-evolution for automated platformer design,” in *EvoGames 2012*, 2012.
- [31] K. Hartsook, A. Zook, S. Das, and M. Riedl, “Toward supporting storytellers with procedurally generated game worlds,” in *IEEE Conference on Computational Intelligence in Games*, 2011.
- [32] D. Lomas, K. Patel, J. L. Forlizzi, and K. R. Koedinger, “Optimizing challenge in an educational game using large-scale design experiments,” in *ACM SIGCHI Conference on Human Factors in Computing Systems*, 2013, pp. 89–98.
- [33] J. Lin, “The science behind shaping player behavior in online games,” 2013, game Developers Conference. [Online]. Available: <http://www.gdcvault.com/play/1017940/The-Science-Behind-Shaping-Player>
- [34] L. Terveen and D. W. McDonald, “Social matching: A framework and research agenda,” *ACM Transactions on Computer-Human Interaction*, vol. 12, no. 3, pp. 401–434, 2005.
- [35] C. Phua, V. C. S. Lee, K. Smith-Miles, and R. W. Gayler, “A comprehensive survey of data mining-based fraud detection research,” *Computing Research Repository*, vol. abs/1009.6119, 2010.
- [36] E. Hastings, R. K. Guha, and K. Stanley, “Automatic content generation in the galactic arms race video game,” *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 1, pp. 245–263, 2009.
- [37] S. Risi, J. Lehman, D. B. D’Ambrosio, R. Hall, and K. O. Stanley, “Combining search-based procedural content generation and social gaming in the petalz video game,” in *8th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2012.
- [38] G. Smith, J. Whitehead, and M. Mateas, “Tanagra: Reactive planning and constraint solving for mixed-initiative level design,” *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 3, no. 3, pp. 201–215, 2011.
- [39] J. Dormans, “Machinations: Elemental feedback structures for game design,” in *GAMEON-NA*, 2009, pp. 33–40.
- [40] S. Amershi, J. Fogarty, A. Kapoor, and D. Tan, “Effective end-user interaction with machine learning,” in *25th AAAI Conference on Artificial Intelligence*, 2011, pp. 7–11.
- [41] J.-y. Hong, E.-h. Suh, and S.-J. Kim, “Context-aware systems: A literature review and classification,” *Expert Systems with Applications*, vol. 36, no. 4, pp. 8509–8522, 2009.
- [42] A. Macvean, S. Hajarnis, B. Headrick, A. Ferguson, C. Barve, D. Karnik, and M. O. Riedl, “WeQuest: Scalable alternate reality games through end-user content authoring,” in *8th International Conference on Advances in Computer Entertainment Technology*, 2011, p. 22.
- [43] A. Gustafsson, J. Bichard, L. Brunnberg, O. Juhlin, and M. Combetto, “Believable environments: generating interactive storytelling in vast location-based pervasive games,” in *ACM SIGCHI International Conference on Advances in Computer Entertainment Technology*, 2006, p. 24.
- [44] M. G. Friberger, J. Togelius, A. B. Cardona, M. Ermacora, A. Moustén, M. Jensen, V. Tanase, and U. Brøndsted, “Data games,” in *4th Workshop on Procedural Content Generation*, 2013.
- [45] C. Grappiolo, J. Togelius, and G. N. Yannakakis, “Interaction-based group identity detection via reinforcement learning and artificial evolution,” in *GECCO Evolutionary Computation and Multi-agent Systems and Simulation Workshop*, 2013.
- [46] M. A. Ahmad, B. Keegan, D. Williams, J. Srivastava, and N. Contractor, “Trust amongst rogues? a hypergraph approach for comparing clandestine trust networks in MMOGs,” in *Fifth International AAAI Conference on Weblogs and Social Media*, 2011, pp. 17–21.
- [47] A. Fujita, H. Itsuki, and H. Matsubara, “Detecting real money traders in MMORPG by using trading network,” in *7th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2011, pp. 26–31.
- [48] M. O. Jackson, *Social and economic networks*. Princeton University Press, 2010.
- [49] E. Law and L. Ahn, “Human computation,” *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 5, no. 3, pp. 1–121, 2011.



# Approaches to Measuring the Difficulty of Games in Dynamic Difficulty Adjustment Systems

Dagmara Dziedzic & Wojciech Włodarczyk

To cite this article: Dagmara Dziedzic & Wojciech Włodarczyk (2018): Approaches to Measuring the Difficulty of Games in Dynamic Difficulty Adjustment Systems, International Journal of Human-Computer Interaction, DOI: [10.1080/10447318.2018.1461764](https://doi.org/10.1080/10447318.2018.1461764)

To link to this article: <https://doi.org/10.1080/10447318.2018.1461764>



Published online: 20 Apr 2018.



Submit your article to this journal 



Article views: 12



View related articles 



View Crossmark data 



# Approaches to Measuring the Difficulty of Games in Dynamic Difficulty Adjustment Systems

Dagmara Dziedzic<sup>a</sup> and Wojciech Włodarczyk<sup>b</sup>

<sup>a</sup>Department of Logic and Cognitive Science, Institute of Psychology, Adam Mickiewicz University, Poznań, Poland; <sup>b</sup>Faculty of Mathematics and Computer Science, Information Systems Laboratory, Adam Mickiewicz University, Poznań, Poland

## ABSTRACT

In this article, three approaches are proposed for measuring difficulty that can be useful in developing Dynamic Difficulty Adjustment (DDA) systems in different game genres. Our analysis of the existing DDA systems shows that there are three ways to measure the difficulty of the game: using the formal model of gameplay, using the features of the game, and direct examination of the player. These approaches are described in this article and supplemented by appropriate examples of DDA implementations. In addition, the article describes the distinction between task complexity and task difficulty in DDA systems. Separating task complexity (especially the structural one) is suggested, which is an objective characteristic of the task, and task difficulty, which is related to the interaction between the task and the task performer.

## 1. Introduction

For years, the difficulty was an integrated element of the game design process (Schweizer, 2006). First attempts to manipulate the difficulty of the game emerged with the launch of first gaming platforms and the coin-op arcade games in the 1970s. It was the time when games began to give the player an opportunity to choose the level of difficulty from one of the predefined options. As mentioned by Schweizer (2006), the first popular video game in which a player had a choice between two options (beginner's race and advanced player's race) was SpeedRace – an arcade game produced in 1974 by the Taito Corporation (1974). Soon after, game developers went one step further and created a first video game in which the difficulty was adjusted to player's doings. This game was the Heart of Africa (1985), in which the key locations were moved if the player had a problem with finding them (Brathwaite & Schreiber, 2008).

Today, the ability to adjust the game's difficulty to the needs and experience of a player is associated with the ability to choose one of several available options (e.g., easy, medium, hard). The choice between these options or their counterparts began to be commonly used in games in the 1990s (e.g., in Sid Meier's Civilization, 1991; Doom, 1993; Jazz Jackrabbit 2, 1998) and is still used today (we can find it in the latest productions, e.g., in Call of Duty, 2003; The Witcher, 2007; Dishonored, 2012; Sid Meier's Civilization VI, 2016). Regardless of which difficulty the player chooses, the difficulty of the game will usually be further increased with the progress – the further the player proceeds, the more difficult the game becomes.

In the examples described earlier, the player decides which difficulty of the game he/she wants to choose, but his/her choice is limited just to few options that were previously prepared by game designers. In addition, the difficulty of the game is always incremented at the fixed point in a game (e.g., when a player reaches a certain level). This approach is an example of the game-centered game design, in which human experts (i.e., designers) independently determine the parameters of the game (see Hristova, n.d.). In this type of game design, difficulty curve is static and is not influenced by the behavior of the player during the game. This approach treats every player equally and ignores differences in their abilities and preferences. This way, in some periods of the gameplay, difficulty may not be matched to the skills of the players (Yun, Shastri, Pavlidis, & Deng, 2009).

Recently, game designers tend to choose an alternative approach – the player-centered game design (or the user-centered game design), in which the games are designed, so that they can meet the expectations of players. One of the essential elements of player-centered game design is to provide an appropriate level of challenge, a properly matched learning curve and an enhancement of the gameplay experience for each player (Charles & Black, 2004). In this approach, developers design the so-called Dynamic Difficulty Adjustment (DDA) systems to ensure proper difficulty for each individual player.<sup>1</sup> Their purpose is to modify some elements of the gameplay, so that the general difficulty will be changed in the response to a player's performance, or his/her affective states. Measurement of difficulty in the DDA is a

**CONTACT** Dagmara Dziedzic  [dagmara.dziedzic@amu.edu.pl](mailto:dagmara.dziedzic@amu.edu.pl)

<sup>1</sup>What is more adjusting the difficulty should be done without interrupting the game, so the player does not feel cheated by the game. Effective difficulty adjustment should take place without degrading player's experience and influencing his/her sense of control and accomplishment. For more about player's enjoyment and satisfaction, see Hunicke (2005) and Andrade, Ramalho, Gomes, and Corruble (2006).

case of measuring the difficulty in which the measurement directly affects the adjustment. The goal is not to measure the difficulty by itself, but rather focus on the issue of how it allows the system to adjust the difficulty of the game to the player.

This approach is very promising, because game design, where the difficulty is adjusted to the player's skills can bring many benefits. The problem of difficulty in the game is important for both practical and theoretical reasons. From a practical point of view, in this model, game developers are able to increase their profits and reach more customers. However, the design of DDA systems applies not only to commercial productions – this approach can be also easily implemented in games with a purpose, serious games, educational games, and rehabilitation games. Games with challenges adjusting to the skills of players can reach a larger audience, be more interesting for the players and educate them more effectively. For example, Hamari et al. (2016) report that educational games in which challenges increases with the growing skills and knowledge of students can efficiently engage them in learning activities.

From a theoretical point of view, the researchers of the difficulty in games (intentionally or not) are trying to answer questions that go beyond the realm of games, such as What is the difficulty? Can the difficulty be defined? How a person feels and perceives the difficulty? How the difficulty can affect the motivation and satisfaction?

Currently, there is no consistent definition of the difficulty that would be used by all designers of DDA systems. There is also no inclusive approach to the problem of difficulty and the possibility of adjusting it. In addition, in existing literature, concepts of difficulty and complexity are not distinguished and often used interchangeably.

This article is an attempt to present existing ways of measuring the difficulty of games for purpose of DDA systems. Within the article, we distinguish three types of difficulty measurement techniques. The DDA designer creating a new system can choose the technique which suits the characteristics of game he/she is working on. Moreover, the article attempts to address the issue of distinguishing the complexity and difficulty of games.

The article describes in [Section 2](#) – purpose of DDA systems, in [Section 3](#) – difficulty measurement techniques in DDA systems, and in [Section 4](#) – the difference between the concepts of game's complexity and game's difficulty.

## **2. Purpose of DDA systems**

As mentioned in [Section 1](#), the implementation of DDA systems is the realization of player-centered game design. Its main goal is to adjust the gameplay to the skills of the individual player. To achieve this, the designers of DDA often rely on the flow theory, which suggests how well-designed automatic difficulty adjustment should work (regardless of the genre of games). Flow concept was

introduced by Csikszentmihalyi (1975) who describes it as a “holistic sensation that people feel when they act with total involvement” (p. 36). Flow is something felt by people who are totally involved in a task they are performing, and this task is done for the pure enjoyment.

The state of flow can be successfully achieved by the players. However, to achieve it, appropriate conditions have to be met. According to Nacke (2012) in the case of games, the player must first perform a challenging activity that requires him/her to improve his/her skills. Second, the activity performed by him/her should have clear and easy to achieve goals (levels, missions, high scores, etc.) with an immediate feedback on his/her progression. Third, the final result of the activity the player is performing should be uncertain, but at the same time, he/she has to have a direct impact on its outcome.

One of the most important conditions for experiencing the flow is to provide the challenges that the player perceives as adequate to his/her abilities. Missura (2015) points out that an integral feature of any challenge (and the learning process required to master it) is its difficulty. The difficulty is a subjective factor, which changes (decreases) with time and effort that the player spends on learning the skill required to accomplish a certain task (Missura, 2015).

The flow theory suggests that the difficulty of the game should offer challenges that difficulty corresponds to the current skills of the player. If the player feels that the challenges in the game become too difficult for him/her too quickly – he/she experiences frustration (Falstein, 2005) or even an anxiety (Csikszentmihalyi, 1975). If the difficulty of the challenges don't change – and the skill of the player rises – the player experiences a boredom (Csikszentmihalyi, 1975). The path between these two extremes is referred to as the flow zone.

One of the solutions, which may be used to ensure that the player stays in the flow zone, regardless of his/her experience and skills is using DDA systems. Depending on the genre of the game, difficulty is handled in a different way, using different parameters of the game.

## **3. Difficulty measurement techniques in DDA systems**

While creating a DDA system, a designer has to define a method for measuring the difficulty and conditions describing how it will be manipulated. After an analysis of existing DDA systems, one may realize that depending on various types of games, the difficulty can be manipulated using different elements of the game, also the intervention of a DDA system (which increases or reduces the overall difficulty of the game) may occur in different points during the game.<sup>2</sup> Therefore, these properties of DDA systems are usually imposed by the structure of the game itself (for more general discussion, see Dziedzic, 2016). Even though there is no single, generally accepted definition of the difficulty, designers are able to

<sup>2</sup>The analysis was carried out with four search engines for scientific articles: Google Scholar (<https://scholar.google.com>), EbscoHost (<https://search.ebscohost.com>), ACM Digital Library (<http://dl.acm.org/>), ScienceDirect (<http://www.sciencedirect.com/>), using a combination of keywords such as *dynamic, difficulty, adjustment, DDA, games*.

create a DDA system using simplified, informal metrics based on the characteristics of the game they are working on.

Examples of the DDA systems created in this way are listed in [Tables 1–3](#). These are examples of measuring and/or adjusting the difficulty of games that allowed us to identify the three general approaches used to measure the difficulty. Each example includes the definition of difficulty proposed by the game authors, difficulty measurement (which describes in detail how the aforementioned definition was implemented), and the way in which difficulty was adjusted (if it was included in the research). Examples of DDA systems outlined in [Tables 1–3](#) not only allowed us to distinguish three approaches to measure the difficulty but also showed that some DDA systems are based primarily on game complexity (see [Section 4](#)).

The difficulty measurement techniques described later define three approaches that can be used while developing a DDA system. The first two are directly related to gameplay, and the third is player-related. The first technique involves using a formal model, which can be defined as a detailed description of rules of the game (one of the consequences of having a formal model of the game is the ability to construct its game tree<sup>3</sup>). The second technique is used when the formal model of the game is unknown (or highly complicated). In such case, DDA systems use the features of the game to create an approximate measurement for the real difficulty. We distinguish two types of features: parameters that describe the selected elements of the game (e.g., number of opponent's pawns, number of enemies, number of lost lives, number of collected coins) and player's performance indicators that do not have direct representation in the game (e.g., the number of enemies killed per minute, average rate of gain of coins). In what follows, the terms "parameters" and "indicators" will be understood as described earlier. Apart from using gameplay-related techniques, creators of DDA systems can independently analyze the player himself/herself. In the third technique of measuring difficulty, DDA systems are evaluating the feeling of the difficulty perceived by the player. The measurement of difficulty involves querying the player about his/her perceived difficulty, or measuring emotions he/she felt during the game. This measurement can take place independently of the gameplay, so it can be mixed with the first two difficult measurement methods.

To sum up, one can define three distinct approaches to measure the difficulty: (1) the use of formal model of gameplay, (2) the use of selected features of the game, and (3) the direct examination of the player.

### **3.1. Using the formal model of gameplay**

The use of this technique occurs in board games such as chess or connect four (see [Table 1](#)). These are games in which the player chooses his/her next move from a finite

set of moves possible in his/her current state of the game and the gameplay is divided into units of time (turns).

Within this group, DDA systems are based on the analysis of the structure of the game tree. The number of possible moves at every moment is relatively small. Performance of the player (whether the player is going to win or lose) is evaluated based on the analysis of his/her movements – and more specifically possible consequences of his/her moves retrieved from the game tree. DDA systems working as described previously are most often based on search algorithms such as MiniMax or Alpha-beta Pruning (see [Missura & G'Artner, 2008](#)). Algorithms of this type operate on the basis of the evaluation function of each player's movement. Typically, this function analyzes the consequences of player's move as a change in the game's parameters before and after player's action (e.g., change in the number of enemy pawns). For example, in chess, capturing opponent's pawn can be considered as a good move which leads to winning, and loss of pawn as a bad move that increases the chances that the player will lose. Using this knowledge, the system changes the current difficulty by selecting opponent's moves which difficulty matches those done by the player (the same measure is used to evaluate both player's and opponent's moves). One of the most important advantages of this approach is that even if game trees of different games can vary a lot, they can all be analyzed in the same manner. Regardless of whether a DDA system is designed for games like Tic-Tac-Toe or chess, it can use evaluation function based on the structure of the game tree such as the shortest path to success, or the average path length to failure. Intervention of the DDA system, created for this type of games, can be done in units of time naturally imposed by the game – after each round. Exemplary systems of the described type are presented in [Table 1](#). In chess by [Guid and Bratko \(2013\)](#) and "Mastermind" by [Gerasimczuk, Der Maas, and Raijmakers \(2013\)](#), authors have proposed only a measure of difficulty, without including a DDA system in their research. However, both solutions show a good example of difficult measurement techniques for games with a formal model. Due to the fact that adjusting difficulty in various games with a game tree can be fairly similar, creating a measure of difficulty in this type of games is more important than creating an adjustment algorithm that could be transferred from one game to another.

Main assumption that one has to accept if he/she chooses to use this approach is that the structure of the game can reflect the performance of the player and influence the way he/she perceives the difficulty (features such as the length of the shortest path to success determines the difficulty of the game).

### **3.2. The use of selected features of the game**

The second technique occurs in games, where the formal model is not known or it is too complicated to use it – due to a large

<sup>3</sup>Game tree refers to the concept of a tree known from the graph theory. In this tree, every possible state of the game is represented as a node in a graph. And if a player can move from a given state to another in exactly one move, then these states are connected with an edge. The final game states are represented as leaves of the tree. Each play of the game is represented as a path leading from the root to one of the leaves (for more details, see [Gobe, De Voogt, & Retschitzki \(2004\)](#)).

**Table 1.** The use of a formal model of the game in exemplary implementations.

| The use of formal model of the game                  |  |  |  |
|--|--|--|--|
| Game title and source                                | Definition of difficulty used in the research  | Difficulty measurement   | Adjustment   |
| Connect Four by Missura and Gärtner (2008)           | "The problem of an <b>automatic difficulty scaling</b> can be viewed then in a context of an interaction between a player and one (or more) in-game agent... It is natural to assume that at any given time the agent has a set of actions (strategies) available to it. The question of how to adjust the game difficulty automatically can then be formulated as what action (strategy) should an in-game agent choose as next." (Missura & Gärtner, 2008, p. 1)   | Authors proposed a search-based algorithm called AdaptiveMiniMax (AMM). <ul style="list-style-type: none"> <li>(1) Using a full game tree, MiniMax algorithm is creating a set of all possible moves together with their evaluation.</li> <li>(2) Score of each move depends on length of its path to winning or losing (it gets more points if player can win faster).</li> </ul>   | (1) Algorithm evaluates moves done by the player using this metric and creates a ranking of them.<br>(2) AMM adjusts the difficulty of the game by selecting moves that matches an average score from the ranking. |
| Chess by Guid and Bratko (2013)                      | "We focus our investigation on problems in which the <b>difficulty</b> arises from the combinatorial <b>complexity of problems</b> . We propose a <b>measure of difficulty</b> that is based on modeling the problem solving effort as search among alternatives and the relations among alternative solutions." (Guid & Bratko, 2013, p. 860)   | Authors proposed an search-based algorithm for measuring difficulty of chess tactical problem. Proposed metric was created using following principles: <ul style="list-style-type: none"> <li>(1) If its solution requires many steps (solution lies deep in the game tree), the problem is considered difficult.</li> <li>(2) The problem is more difficult if the possible solutions are located far from each other in the game tree.</li> </ul>  | N/A  |
| Mastermind (simplified) by Gerasimczuk et al. (2013) | "We associate the <b>difficulty</b> of Deductive Mastermind <b>game-items with the size</b> of the corresponding logical trees obtained by the tableau method." (Gerasimczuk et al., 2013, p. 297)<br><br>"Normatively speaking, the full tree generated by the tableau method for the set of formulas corresponding to a DMM-item, represents an adequate reasoning scheme. Therefore the <b>size of the tree</b> (e.g. the number of nodes) can be thought of as an <b>abstract complexity measure</b> ." (Gerasimczuk et al., 2013, p. 307) | The authors developed a model for calculating the difficulty of the game, which is based on the logical reasoning, using analytic tableaux. <ul style="list-style-type: none"> <li>(1) The algorithm contains a list of logic formulas that are used for reasoning. Formulas describe basic conclusions that can be made from the game's feedback.</li> <li>(2) The algorithm creates a tree (similar to game tree) with the states, which can be reached using the logical reasoning.</li> <li>(3) The difficulty of the game is computed as a size of the tree.</li> </ul> | N/A  |

number of moves the player can make it is not possible to create a full game tree. Games of this type often include arcade elements, random events or time-related tasks. Examples of games using this technique belong to such genres as first-person shooters, tower defense, platformers, etc. (see Table 2).

Because it is not possible to determine all the possible moves of the player, in this method, the model of the difficulty is reduced only to a closed group of features. In this model, DDA can use both parameters that describe the selected elements of the game (e.g., number of enemies) and indicators of player's performance (e.g., number of enemies killed per minute). Because these features are very dependent on the game's genre and the gameplay elements, each game has its own unique set of features, manually selected by the DDA designer.

Also, the intervention time of the system depends strictly on the characteristics of the game for which DDA is designed. If it is a game with distinct stages of gameplay (e.g., waves of enemies in tower defenses), the intervention occurs after each stage. Otherwise, if the game does not contain such stages, the

intervention must occur in some fixed intervals (e.g., every 30 s).

DDA systems using this technique are based on various algorithms. Recently, one of the most popular approaches is to use machine learning algorithms (such as clustering algorithms – see Lora, Sánchez-Ruiz, González-Calero, & Gómez-Martín, 2016) to determine the relation between players performance and the difficulty of the game.<sup>4</sup> Exemplary systems, which use this technique of measuring the difficulty, are described in Table 2.

Main assumption that one has to accept if he/she wants to use this approach is that selected features are adequate and sufficient to measure and manipulate the difficulty felt by the player.

### 3.3. Direct examination of the player

In this method, designers of DDA systems analyze the difficulty perceived by the player instead of elements of the gameplay. Therefore, this approach can be applied to any game,

<sup>4</sup>Machine learning algorithms are designed so that for a set of input features algorithm assigns a result value (numerical for regression algorithms or class for classification algorithms). Machine learning works well in this approach because the difficulty model is also based on a set of features. A DDA creator using machine learning must prepare a training data with specific values of the features and a specific difficulty level assigned to them. Then the learning algorithm will try to find the relationship between the features and the level of difficulty. One of the most popular uses of machine learning in games is creating an artificial intelligence (AI) that plays with the player (e.g., AlphaGo created by DeepMind, <https://deepmind.com/>). While AI algorithms mentioned earlier is designed to simply win with the player, the purpose of the DDA is to adapt to his/her performance. For more details about machine learning, see Smola and Vishwanathan (2008).

**Table 2.** The use of selected parameters of the game in exemplary implementations.

| The use of selected parameters of the game                      |   |  |   |
|---|---|--|---|
| Game title and source   | Definition of difficulty used in the research   | Difficulty measurement   | Adjustment  |
| Hamlet (first-person shooter) by Hunicke and Chapman (2004)     | "Using techniques drawn from Inventory Theory and Operations Research, Hamlet <b>analyzes and adjust the supply and demand of game inventory</b> in order to control overall <b>game difficulty</b> ." (Hunicke & Chapman, 2004, p. 96)<br>"We propose a probabilistic technique that dynamically evaluates the <b>difficulty</b> of given obstacles based on user performance, as the game is running." (Hunicke & Chapman, 2004, p. 97)   | Authors proposed a probabilistic model for measuring difficulty in the game using player's inventory. The algorithm uses the following rules to determine the current performance of a player.<br>(1) Algorithm observe player's health and distribution of damages taken by player over time to predict potential health shortfalls.<br>(2) Algorithm analyzes changes in player's inventory. How much he gains (e.g., ammunition found in creates) and losses during the game (e.g., used ammunition). | Authors created a number of adjustment policies with adjustment actions and cost estimations.<br>(1) System support two types of actions: reactive (adjusting elements that are "on stage", e.g., entities that have notices the player) and proactive (adjusting elements that are "off state" e.g., spawning health).<br>(2) System is determining the cost of a given action to make sure that player do not notice system's intervention. |
| Tower Defense by Sutoyo, Winata, Oliviani, and Supriyadi (2015) | "... we determine the <b>difficulties</b> based on players' <b>lives, enemies' health, and passive skills</b> ... that are chosen by the player. With three of these factors, players will have varies experience of playing tower defence because different combination will give different results to the system and difficulties of the games will be different for each gameplay." (Sutoyo et al., 2015, p. 435)  | Author selected three parameters that are monitored to determine difficulty of the game through player's performance.<br>(1) Algorithm analyzes player's lives, enemies' health and passive skills (skill points).<br>(2) Passive skills are three groups: offensive, defensive and support.   | (1) DDA modifies the set of parameters (such as the number of gold or spawned opponents) and their multipliers (e.g., gold multiplier).<br>(2) The DDA system adjusts the difficulty by modifying the aforementioned parameters after each round (e.g., decreases the number of opponents).   |
| Tetris by Lora et al. (2016)                                    | "When a player starts a new game we look at his first movements to find the most similar cluster. Then the system provides dynamic help to the player choosing "good" Tetris pieces from time to time. In our experiments, using DDA users obtain higher scores and report improvements in terms of their game experience." (Lora et al., 2016, p. 335)<br>"The <b>difficulty</b> in Tetris depends mainly on two parameters: The <b>type of pieces</b> and the <b>falling speed</b> ." (Lora et al., 2016, p. 337) | The authors created a statistical clustering model that is used to estimates player's skills.<br>(1) Performance of the player is calculated based on his/her last 10 tactical decisions (placing the block on the map).<br>(2) Each decision is described by a set of parameters (e.g., height of the blocks, scored points, maximum number of points to win).<br>(3) Using collected data players were divided into three groups: newbie, average, expert.   | (1) Algorithm is assigning player to a one of aforementioned groups (algorithm is updated after each move).<br>(2) Depending on player's group, algorithm helps him/her frequently (for newbie) or infrequently (for experts). Algorithm helps the player by giving him block needed in current state of the game.  |

regardless of whether it has a formal model or not (see Table 3).

The difficulty measure of the game is based on the player's feelings. The specific feelings of a player are correlated with the various difficulty settings of the game. When a measuring tool recognizes that the game is too hard for the player (or too easy), the DDA system adjusts the difficulty.

Just like in the previous case, this technique does not impose the moment of intervening – as it depends directly on the characteristics of the game.

Examples of methods used for measuring the difficulty are questionnaires measuring affective states and physiological tests. Example systems, which use this technique of measuring the difficulty, are described in Table 3. All the examples cited in this table apply only to research carried out with the use of games, and not commercial games. Even though it is hard to apply this technique in commercial projects, it can provide the most reliable information about the perceived difficulty directly from the player. Furthermore, it is possible to use this technique to assign difficulty to appropriate gameplay parameters (e.g., the number of enemies) based on difficulty perceived by the player. At the end of the study, collected data can be used to create a statistical adjustment model that does

not require polling (or testing) the player, and may already be used in commercial games.

Main assumption that one has to accept if he/she chooses to use this approach is that the player is able to determine the difficulty of the task and compare it with the difficulty of other, and that selected affective or physiological states are directly related to perceiving the difficulty of the game.

### 3.4. Conclusions

Due to the fact that it is difficult to find a game where it is possible to implement all three techniques of measuring difficulty in DDA systems and to test which gives better results, it is impossible to decide which of the above could be consider as a best one. Choosing the right technique highly depends on the structure of the gameplay and the purpose of the DDA system. Table 4 lists the practical suggestions that can help to select the right difficulty measurement technique for a new DDA system. It provides a summary of approaches described in this section, their advantages, disadvantages, and the suggested types of games for each of the techniques.

**Table 3.** Direct examination of the player in exemplary implementations.

| Direct examination of the player                |  |  |   |
|---|--|--|---|
| Game title and source                           | Definition of difficulty used in the research  | Difficulty measurement   | Adjustment  |
| Third-person shooting game by Yun et al. (2009) | "The measurements are based on the assumption that the players' performance during the game-playing session alters blood flow in the supraorbital region, which is an indirect measurement of increased mental activities." (Yun et al., 2009, p. 2195)<br>"... we propose a system that adaptively and automatically adjusts the <b>game difficulty level</b> based on measurements of the <b>players' facial physiology</b> ." (Yun et al., 2009, p. 2196) | Authors proposed a method for measuring difficulty by analyzing the player's changes in the skin temperature in the supraorbital region using the StressCam.<br>(1) The change in temperature in this region is associated with the blood flow, which is an indirect measure of the mental activities.<br>(2) When the game is hard and the player suffers stress or frustration – the temperature in the supraorbital region should grow. | (1) Authors created three game modes: easy, moderate, difficulty. Each mode is associated with different strength of enemies robots.<br>(2) Game difficulty was altered during the game using the metric described earlier.   |
| Pong by Liu, Agrawal, Sarkar, and Chen (2009)   | "In this DDA mechanism, a <b>player's physiological signals</b> [e.g. skin conductance, skin temperature, heart sound] were analyzed to infer his or her probable <b>anxiety level</b> , which was chosen as the target affective state, and the <b>game difficulty level</b> was automatically adjusted in real time as a function of the player's affective state." (C. Liu et al., 2009, p. 506)  | The authors created an affect based DDA, which recognize the affective state (anxiety) using psychophysiological analysis.<br>(1) To develop affective model, the authors built mappings from physiological features to the intensity of anxiety.<br>(2) Low anxiety means that the player perceives the game as easy and high anxiety level means that the game is to hard.   | (1) Real-time player's anxiety level recognition was used to adjust the game's difficulty.<br>(2) To manipulate the difficulty level of the game, authors selected a number of parameters (e.g., ball speed and size, paddle speed and size, sluggish or over-responsive keyboard, random keyboard) and adjusted them during the game.                                  |
| Runner Game by Medeiros and Medeiros (2014)     | "... we propose a way to automatically rank all combinations of features according to how fun that level is, and by doing so, choose the combination of features with the best flow for our runner game. After each 30 second playthrough, the <b>player is asked</b> to rate <b>how fun</b> and <b>how difficult</b> was that level on a scale from 1 to 5." (Medeiros & Medeiros, 2014, p. 797)  | Authors proposed a system in which the difficulty measurement is done by the player himself after each round of the game.<br>(1) Player plays the game for 30 seconds or until avatar's death.<br>(2) After the game, he/she completes a survey, in which (on a scale from 1 to 5) he/she describes how fun and difficult the game was.  | (1) The authors defined several features of the game (e.g., number of spikes per wave, distance between spikes) that are used to control the difficulty of the game.<br>(2) The system selects a set of features for each game. Every time the player rated a particular configuration to be good/bad – the algorithm changes the probability of current configuration. |

#### 4. Complexity versus difficulty

Analyzing at how the difficulty is defined in games with a full game model (Table 1), one may notice that it is modeled using only game's parameters and actions related to them (e.g., capturing opponent's pawn as a decrease in the number of opponent's pawns on the map). In particular, the difficulty of these games is measured mainly on the basis of their game tree – core of DDA systems do not take into account the indicators of player's performance (how quickly he/she puts the pawns, how logically he/she reasons), nor his/her emotions and physiological states.

If so, the difficulty measurement for games with a full game model corresponds to struturalist viewpoint of task complexity proposed by Liu and Li (2012). These authors provided a comprehensive review and conceptualization of task complexity and tried to determine the differences between task complexity and task difficulty.

Further in this section we rely primarily on their work because it is a holistic overview based on the analysis of several dozen research studies in which the task complexity appeared. Their approach has not been previously applied to games, but in our opinion the complexity of the game can be considered as a special case of much more general task complexity – and more specifically task complexity in structuralist viewpoint.

According to Liu and Li (2012), in structuralist approach, task complexity is defined as the structure of the tasks and it may be computed as a function of the number of elements that form the task and the relations between these elements (e.g., number of elements, number of targets, number of solutions, number of tracks, number of alternatives).

In the case of the complexity and the difficulty, there are no definitions which would clearly explain these concepts, as a consequence, they are often confused. In accordance with Liu and Li (2012), we claim that in the case of games:

To avoid the misuse and misunderstanding of these related terms, we claim that task complexity can be distinguished from task difficulty in that task complexity involves the objective characteristics of a task, whereas task difficulty involves the interaction among task, task performer, and context characteristics. Subjective task complexity is a perceived complexity by the task performer. In general, task difficulty refers to the extent to which task performers feel difficulty in performing a task. (p. 559)

The term difficulty includes not only the complexity of the task but also the performer who performs it and the context of the task. If so, then the complexity of the game is only a subset of the difficulty, and to create a more complete game difficulty model for games with a full model, DDA designers should consider incorporating the aspect of the task performer (player) into their systems.

DDA systems created for games with a full model operate on the basis of objective gameplay characteristics and they depend only on actions of a player (and not his/her skills or emotions). Liu and Li (2012) suggested that objective task complexity is independent of task performer and it is closely related to task characteristics. By looking at the definitions in Table 1 (compared to Tables 2 and 3) only in the first one, the authors define the difficulty based solely on the task complexity. The definitions imposed by the authors are determined only by the structure of the problem that player has to solve (as in the structuralist task complexity by Liu & Li, 2012). In this approach, the role of the performer is limited or does not appear at all. In addition, taking the performer into account can be a challenge, because in logic games it is difficult to capture concepts such as logical reasoning (which affects the next move of the player) in a measurable way. In the game with a full game model, DDA system analyzes only the final effect of the player's actions (e.g., the fact of capturing the pawn on the board in chess is more important than the number of pawns captured per minute).

In the games where the formal model of the game is unknown (or highly complicated), complexity is also present but it cannot be described in a simple way (e.g., due to the limited computing power of current computers). Hence, it is necessary to simplify the game to a closed set of measurable features. In games of this type, DDA system analyzes the sequence of player's actions spread over time (e.g., the time needed for the player to aim at the enemy). The task performer's aspect is measured indirectly with the player's performance indicators, which directly reflect the player's skill (such as hand-eye

coordination). The task difficulty seems to be more appropriate for describing a game in which the task complexity is not determined by the formal model and the aspect of the task performer is taken into account.

In the case where the task performer's aspect is measured directly with the player's examination, the presence of the performer's task is evident – regardless of whether the game contains a full model or not, this technique is based on task difficulty. A summary of what constitutes the base of the difficulty measure for each difficulty measurement technique is given in Table 4.

The conclusion that complexity is only a subset of difficulty can also be reached by considering situations in which the same complexity of tasks may have different difficulties for different performers. For example, for most children learning the multiplication tables, multiplying "six times eight" is far more difficult than the multiplication of other numbers. Multiplying "six times ten" and "six times eight" have the same structural complexity, but in the latter case, the children respond poorly in 62.5% of cases (Magazine Monitor, 2014).

Another case, in which the same structural complexity of the task gives a different feeling of the difficulty is chess. In studies conducted by Hristova, Guid, and Bratko (2014), chess experts evaluated the difficulty of chess tactical problems. For this purpose, authors selected 12 positions – chess tactical problems (selected randomly from Chess Tempo<sup>5</sup>), which were divided into three classes of difficulty (easy, medium, hard). This division was made on the basis of the Chess Tempo rating – ranking, which is based on the success and failures of the players (if the player solves the problem correctly, its rating goes down, and if incorrectly –

**Table 4.** Comparison of difficulty measure techniques in DDA systems.

| Technique name                              | The use of formal model of gameplay  | The use of selected features of the game  | The direct examination of the player  |
|---|--|---|---|
| Difficulty measure                          | Based on the game tree   | Based on the game parameters and player's performance indicators  | Based on the difficulty perceived by the player   |
| Base of difficulty measure                  | Task complexity (task performer's aspect is limited or does not appear)                          | Task difficulty (task performer's aspect is measured indirectly with player's performance indicators)   | Task difficulty (task performer's aspect is measured directly with the player's examination)  |
| Difficulty adjustment method                | Changes the difficulty by selecting moves of the opponent using a measure based on the game tree | Changes the parameters of the game affecting the difficulty   | Changes the parameters of the game affecting the difficulty   |
| Advantages                                  | DDA systems created for different games share similar algorithms (e.g., MiniMax)                 | The possibility to create DDA systems for complex games by narrowing them to a set of measurable features   | The player directly reports his or her feelings of difficulty rather than indirectly, through the values of the selected metrics  |
| Disadvantages                               | Requires a full game model   | Large range of parameters and indicators that are the basis of DDA systems for different games (there can be multiple, different DDA systems created for the same game) | Creators of DDA system need an access to the player (large interference in the player's environment) and to have the tools to measure the player's, e.g., declared difficulty, affective states or physiological states |
| Typical moment of DDA system's intervention | After each game's turn   | Depends on the structure of the game (the moment is chosen by the designer or after each turn in the game)  | Depends on the moment when the player is questioned and the game structure (the moment is chosen by the designer or after each turn in the game)  |
| Suggested game types                        | Board games, logic games   | Arcade games, shooter games, racing games, platformers  | Arcade games, shooter games, racing games, platformers  |

<sup>5</sup>The Chess Tempo is an online chess platform available at [www.chesstempo.com](http://www.chesstempo.com).

its rating goes up). Chess experts were asked to solve these 12 tactical problems, then rate them from 1 to 12 (they did not know that the problems had previously been divided into three classes of difficulty using the Chess Tempo rating). The study included a retrospective reports, which would help to understand the approach that experts used while solving a certain position. The results showed a large disproportion between Chess Tempo rating, and the estimations of experts. Sometimes, the difficult chess problem was considered easy by the experts and easy problem as hard. The same arrangement of pieces on a chessboard (structural complexity) produced a different result in the assessment of the difficulty made by the experts.

The same complexity of gameplay can result in different difficulty for different players. Therefore, DDA systems that do not take into account aspects of the player will not be able to fully adapt to the specific player. The problem of dependency between complexity and difficulty in games is still open and there is currently no research showing how these two concepts affect each other.

## 5. Conclusions

For years, designers have created games that give gamers a choice of gameplay difficulty. In the most basic approach, players can choose between easy, medium, and hard mode of game. However, this may be insufficient because the players' skills and experience are so different that it is difficult to grasp them just with the use of a finite number of categories. In addition, the player learns during the game, so his/her skills and expectations of the difficulty may change over time. The solution to this problem is to design DDA systems that are based on the flow theory. According to it, the challenge must be adjusted to the player's skill, so he/she can constantly enjoy the game. DDA systems are designed to measure difficulty (based on their difficulty conception), and adapt it.

However, existing DDA systems are missing a coherent approach to measure the difficulty. After an analysis of existing implementations of DDA systems, we have proposed a difficulty measurement techniques, which can be applied in different game genres. The approaches presented in this article distinguish three types of difficulty measures: using the formal model of gameplay, using the features of the game, and direct examination of the player. The first two are separable and relate to gameplay itself, while the third one is related only to the player and can be used independently of the first two (together with them or completely separate).

The designer creates a DDA system for full model games, and it should definitely use it to adjust the difficulty. But taking into account conclusions from Section 4, he/she may consider extending the difficulty model and including the performer's performance (e.g., by incorporating the techniques of measuring difficulty from direct examination of the player). If the designer creates a DDA for a game without a full model, then he/she should use a closed group of game parameters along with indicators of player's performance to adjust the difficulty. The selection of features depends on the game genre and gameplay elements. As basic features, it is important to choose the ones that define the key elements of the gameplay: e.g., in the

shooter games – the number of killed enemies, in tower defence – the number of lives, and in racing games – speed of the car and player's ranking position.

In addition, this article describes the distinction between task complexity and task difficulty in DDA systems. We suggest separating task complexity (especially the structural one), which is an objective characteristic of the task, and task difficulty, which is related to the interaction between the task and the task performer. In DDA systems, which use the full model of gameplay (e.g., chess), the adjustment mechanism is based on the complexity of the gameplay (usually, the structure of the game tree, which is an objective characteristic of gameplay). Since this issue (to our best knowledge) has not been noticed before, there is currently no research that would show the influence of task complexity on the perceived (subjective) difficulty or enjoyment felt by player.

## References

- Andrade, G., Ramalho, G., Gomes, A. S., & Corruble, V. (2006). Dynamic game balancing: an evaluation of user satisfaction. In J. E. Laird & J. Schaeffer (Eds.), *AIIDE* (pp. 3–8). Menlo Park, CA: The AAAI Press.
- Arkane Studios. (2012). *Dishonored* [PC game]. Bethesda Softworks.
- Brathwaite, B., & Schreiber, I. (2008). *Challenges for game designers* (1st ed.). Rockland, MA, USA: Charles River Media, Inc.
- CD Projekt RED. (2007). *The Witcher* [PC game]. Atari, CD Projekt.
- Charles, D., & Black, M. (2004). *Dynamic player modelling: A framework for player-centered digital games*. Proceedings of the international conference on computer games: Artificial intelligence, design and education (p. 29–35). Reading: Microsoft Campus.
- Csikszentmihalyi, M. (1975). *Beyond boredom and anxiety*. San Francisco, CA: Jossey-Bass Publishers.
- Dziedzic, D. (2016). Dynamic difficulty adjustment systems for various game genres. *Homo Ludens*, 9(1), 35–51.
- Epic MegaGames. (1998). *Jazz Jackrabbit 2* [PC game]. Gathering of Developers, Logicware, Project Two Interactive.
- Falstein, N. (2005). Understanding fun - the theory of natural funativity. In S. Rabin (Ed.), *Introduction to game development*. Hingham, MA: Charles River Media.
- Firaxis Games. (2016). *Sid Meier's Civilization VI* [PC game]. 2K Games.
- Gerasimczuk, N., der Maas, H. L. J. V., & Rajmakers, M. E. J. (2013). An analytic tableaux model for deductive mastermind empirically tested with a massively used online learning system. *Journal of Logic, Language and Information*, 22(3), 297–314. doi:10.1007/s10849-013-9177-5
- Gobe, F., de Voogt, A., & Retschitzki, J. (2004). *Moves in mind: The psychology of board games*. Hove, East Sussex: Psychology Press.
- Guid, M., & Bratko, I. (2013). *Search-based estimation of problem difficulty for humans*. Artificial intelligence in education - 16th international conference, AIED (pp. 860–863). Memphis, TN. doi: 10.1007/978-3-642-39112-5\_131
- Hamari, J., Shernoff, D. J., Rowe, E., Coller, B., Asbell-Clarke, J., & Edwards, T. (2016). Challenging games help students learn: An empirical study on engagement, flow and immersion in game-based learning. *Computers in Human Behaviour*, 54, 170–179. doi:10.1016/j.chb.2015.07.045
- Hristova, D. (n.d.). *Dynamic difficulty adjustment (DDA) in first person shooter (FPS) games*. Retrieved April 05, 2017 from [https://www.academia.edu/4921794/Dynamic\\_difficulty\\_adjustment\\_DDA\\_in\\_First\\_person\\_shooter\\_FPS\\_games](https://www.academia.edu/4921794/Dynamic_difficulty_adjustment_DDA_in_First_person_shooter_FPS_games)
- Hristova, D., Guid, M., & Bratko, I. (2014). Assessing the difficulty of chess tactical problems. *International Journal on Advances in Intelligent Systems*, 7(3), 728–738.
- Hunicke, R. (2005). *The case for dynamic difficulty adjustment in games*. Proceedings of the 2005 ACM SIGCHI international conference on advances in computer entertainment technology (pp. 429–433). New York, NY: ACM.

- Hunicke, R., & Chapman, V. (2004). AI for dynamic difficulty adjustment in games. *Challenges in Game Artificial Intelligence AAAI Workshop* (pp. 91–96). San Jose, CA.
- id Software. (1993). *Doom [PC game]*. GT Interactive.
- Infinity Ward. (2003). *Call of Duty [PC game]*. Activision, Aspyr.
- Liu, C., Agrawal, P., Sarkar, N., & Chen, S. (2009). Dynamic difficulty adjustment in computer games through real-time anxiety-based affective feedback. *International Journal of Human-Computer Interaction*, 25(6), 506–529. doi:10.1080/10447310902963944
- Liu, P., & Li, Z. (2012). Task complexity: A review and conceptualization framework. *International Journal of Industrial Ergonomics*, 42(6), 553–568. doi:10.1016/j.ergon.2012.09.001
- Lora, D., Sánchez-Ruiz, A. A., González-Calero, P. A., & Gómez-Martín, M. A. (2016). *Dynamic difficulty adjustment in Tetris*. Proceedings of the twenty-ninth international florida artificial intelligence research society conference, FLAIRS (pp. 335–339). Key Largo, FL.
- Magazine Monitor. (2014). *Who, what, why: Why does the sum 7 × 8 catch people out?* Retrieved April 05, 2017, from <http://www.bbc.com/news/blogs-magazine-monitor-28143553>
- Medeiros, R. J. V. D., & Medeiros, T. F. V. D. (2014). *Procedural level balancing in runner games*. Proceedings of the 2014 Brazilian symposium on computer games and digital entertainment (pp. 109–114). Washington, DC: IEEE Computer Society.
- Missura, O. (2015). *Dynamic difficulty adjustment* (Doctoral dissertation). University of Bonn. Retrieved April 05, 2017, from <http://hss.ulb.uni-bonn.de/2015/4144/4144.pdf>
- Missura, O., & Gärtner, T. (2008). *Online adaptive agent for connect four*. Proceedings of the fourth international conference on games research and development cybergames (pp. 1–8).
- MPS Labs. (1991). *Sid Meier's Civilization [PC game]*. MicroProse.
- Nacke, L. E. (2012). *Flow in games: Proposing a flow experience model*. Proceedings of the workshop on conceptualising, operationalising and measuring the player experience in videogames at fun and games (p. 104–108). Toulouse, France: ACM.
- Ozark Softscape. (1985). *Heart of Africa [Commodore 64 game]*. Electronic Arts.
- Schweizer, B. (2006). Difficulty. In H. Lowood & R. Guins (Eds.), *Debugging game history: A critical lexicon* (pp. 109–117). Cambridge, MA: MIT Press.
- Smola, A. J., & Vishwanathan, S. (2008). *Introduction to machine learning*. Cambridge, MA: Cambridge University Press. Retrieved from <http://alex.smola.org/drafts/thebook.pdf,/bib/smola/smola2008ml/thebook.pdf>
- Sutoyo, R., Winata, D., Oliviani, K., & Supriyadi, D. M. (2015). Dynamic difficulty adjustment in tower defence. *Procedia Computer Science*, 59 (1), 435–444. doi:10.1016/j.procs.2015.07.563
- The Taito Corporation. (1974). *Speed Race [Arcade game]*. Author.
- Yun, C., Shastri, D., Pavlidis, I., & Deng, Z. (2009). *O'game, can you feel my frustration?: Improving user's gaming experience via StressCam*. Proceedings of the SIGCHI conference on human factors in computing systems (pp. 2195–2204). New York, NY: ACM. doi: 10.1145/1518701.1519036

## About the Authors

**Dagmara Dziedzic**'s research is associated with the possibility to make games used in the study, which were a useful tool for researchers and an attractive entertainment for the player. She is also working on the use of user experience techniques to improve gameplay enjoyment.

**Wojciech Włodarczyk** focuses on designing and creating crowdsourcing systems used for acquiring linguistic resources. His scientific interests are also connected with creating artificial intelligence agents in computer games, as well as applying machine learning mechanisms in order to improve them.

# DYNAMIC DIFFICULTY ADJUSTMENT IN VIDEO GAMES

## Title

Empirical evaluation of player experience using a machine-learning approach to dynamic difficulty adjustment in video games.

## Author names and affiliations

Nigel Robb<sup>a</sup>, Bo Zhang<sup>b</sup>

<sup>a</sup> University of Tokyo.

<sup>b</sup> East China Normal University.

## Corresponding author

Nigel Robb

Center for Global Communication Strategies, University of Tokyo, Japan

[nigelrobb@g.ecc.u-tokyo.ac.jp](mailto:nigelrobb@g.ecc.u-tokyo.ac.jp)

*Empirical evaluation of player experience using a machine-learning approach to dynamic difficulty adjustment in video games.*

*Abstract*

Dynamic difficulty adjustment (DDA) in video games involves altering the level of challenge provided based on real-time feedback from the player. Some approaches to DDA use measurements of player performance, such as success rate or score. Such performance-based DDA systems aim to provide a bespoke level of challenge to each player, so that the game is neither too hard nor too easy. Previous research on performance-based DDA shows that it is linked to better player performance, but finds mixed results in terms of player experience (e.g., enjoyment). Also, while the concept of flow is regarded as an important aspect of video game experience, little research has considered the effects of performance-based DDA on flow. We conducted an experiment on the effects of performance-based DDA on player performance, enjoyment, and experience of flow in a video game. DDA was achieved using a generalised algorithm. 221 participants played either the DDA version of the game, a control version (difficulty remained constant), or an incremental version (difficulty increased regardless of performance). Results show that the DDA group performed significantly better. However, there were no significant differences in terms of enjoyment or experience of flow.

*Keywords*

video games; dynamic difficulty adjustment; game balancing; flow; performance; adaptive software

## 1 Introduction

Most video games involve some challenge for the player. Some games are generally easy, some are generally hard, and almost all games feature some change in the difficulty level over time; typically, games get harder the further the player progresses (Chang, 2013). In this paper, we refer to this traditional approach as “incremental difficulty adjustment”. Furthermore, a diverse range of people (e.g., in terms of age, gender, disability, motivations, and preferences) play games (Entertainment Software Association, 2017; Williams et al., 2008). Taken together, these points begin to illustrate the complex challenges involved for the game designer when determining the difficulty of a game, to enhance the experience for a range of players. If the game is too easy, more skilled players may be bored; but if it's too hard, less skilled players may be frustrated (Leiker et al., 2016). It is likely that this applies regardless of the genre or intended purpose of the game. Whether it is a fast-paced action game intended to entertain, a puzzle game for mathematics education, a cognitive training game for children with cognitive impairments, or even a language-learning application with game-like features, it is obviously essential that players engage optimally with the software to ensure the desired outcome. As such, the level of challenge provided by a game is an important consideration.

Within this broad issue, one interesting potential solution to some of these challenges lies in dynamic difficulty adjustment (DDA). DDA in video games refers to any technique in which

## DYNAMIC DIFFICULTY ADJUSTMENT IN VIDEO GAMES

the difficulty of the game is altered during the game (or perhaps between games) in response to some feedback about the player's experience (Xue, et al., 2017). The aim is to continuously tailor the difficulty of the game to each individual player. DDA has featured in video games since at least 1981 (Adams, 2008). The promise of DDA lies in the fact that the game designer does not need to pre-determine one specific difficulty curve (or even a range of pre-determined curves, as in games which let the player select, e.g., Easy, Medium, or Hard mode). Instead, the designer can effectively provide a range of possible difficulties which are dynamically selected for each individual player based on their experience of the game. Essentially, this provides a bespoke difficulty curve for each player.

### *1.1 Approaches to dynamic difficulty adjustment*

One important distinction when using DDA is based on the metric used to provide feedback for game adaptation. In this paper, we have so far (intentionally) characterised this feedback very broadly, as "player experience". In practice, player experience can be determined in a variety of ways. We can broadly categorise approaches to DDA in two ways, depending on whether they use players' in-game performance to provide the feedback, or use some information about the players affective state. A combination of these approaches would of course also be possible.

Performance-based DDA involves measuring the player's performance in the game, and adjusting the level of challenge provided accordingly. For example, the survival horror game Left 4 Dead generates a unique experience for individual players by tailoring – among other things – the enemies encountered by the player, based upon measurements of player performance (Booth, 2009). In other words, as the player performs better, the game gets harder, and vice versa. Previous research on performance-based approaches to DDA demonstrates the wide range of choice available in the design of such systems, both in terms of the indicator of player skill (i.e., the feedback), and the game features that are subsequently adjusted. Regarding the measurement of player skill, previous approaches have used, for example, the time taken to complete a task (Sharek & Wiebe, 2015), players' scores (Bateman et al., 2011), or, in more complex systems, multiple measurements may be combined and evaluated to determine the current state of the player (Hunicke, 2005). Regarding the game features that are adjusted, these range from simple adjustments such as changing the speed of the game (Alexander et al., 2013) or changing the number of objects a player must interact with (Nagle et al., 2015; Robb et al., 2019), to more complex systems which alter the behaviour or characteristics of computer-controlled enemy characters (Hunicke, 2005) or dynamically generate the layout of the game environment (Shaker et al., 2010).

Affective DDA refers to any approach which aims to use some feedback about the player's emotional state as the basis for the adaptivity. A growing body of research has investigated the feasibility of using psychophysiological measurement to provide an index of players' emotions during gameplay, and adapt the game experience based on these measurements. Measures used include cardiovascular (e.g., heart rate), electro-dermal activity (i.e., galvanic skin response, which is directly dependent on sweat-gland activity), electromyography (which measures muscle movements), and neuroimaging techniques. The latter category includes techniques such as electroencephalography and functional near-infrared spectroscopy; both of which use sensors attached to the head to provide real-time measurements of brain activity with a relatively high temporal resolution (Thibault et al.,

2016). These techniques have been used to adapt game difficulty, for example, by increasing the speed of the game or decreasing the size of targets. In addition, some studies have used affective feedback to alter other features of a game not related to difficulty, such as lighting and audio effects. For a comprehensive review of research in this area and references for all examples discussed in this paragraph, see Bontchev (2016). One obvious disadvantage of affective-based DDA is the requirement for additional equipment (which is often large and expensive) to obtain the psychophysiological feedback. Therefore, affective DDA is most likely not yet suitable for widespread use in video games. However, technological advances will undoubtedly address this issue. For example, preliminary work shows the potential of using machine vision techniques to infer players' emotional states based on real-time data obtained from the front-facing camera in smartphones and tablets (Bevilacqua et al., 2015; Bevilacqua et al., 2016). However, due to this current limitation of affective DDA, we will focus on performance-based DDA in the remainder of this paper. We will, however, return to the issue of affective DDA in Section 4, and show how the system used in the present study may also be used with affective feedback.

### *1.2 Effects of performance-based dynamic difficulty adjustment*

Previous research has investigated the effects of using performance-based DDA on various aspects of the game playing experience. Several studies in this area have considered the relationship between DDA and player enjoyment. Alexander et al. (2013) used an experimental game to investigate how DDA compared with incremental difficulty adjustment. They found that, while casual gamers reported enjoying a simple 2D game more when the difficulty was dynamically adjusted according to their performance, experienced gamers (who made up most of the sample) enjoyed the game more when the difficulty was adjusted incrementally. This study also showed that players enjoyed the game more when the difficulty was tailored to their gaming experience (casual vs experienced) rather than their performance. However, in a sample of 90 players, only 19 were categorised as casual players. Furthermore, this classification was determined by the players' response to the question "are you a casual or experienced gamer?"; it is therefore difficult to determine how to understand the distinction between casual and experienced gamers as the classification criteria are not explicit.

Nagle et al. (2015) also found that performance-based DDA led to lower player enjoyment than an alternative system in which players could control the level of difficulty throughout the game themselves. They created a 3D game in which players had to memorize a list of objects and locations, then find the objects and place them in the correct location. The difficulty of the game was determined by the number of objects (more is harder) and the number of times they could view the list of objects and location numbers (fewer is harder). However, while player enjoyment was lower with DDA, DDA was associated with better player performance. That is, when they allowed players to control the number of objects and number of times the list could be consulted, performance was significantly lower than when these values were automatically adjusted based on player performance.

Sharek & Wiebe (2015) also showed that DDA was associated with better player performance. Using an isometric puzzle game, they created over 100 different levels which were tested and ordered by difficulty. They had three difficulty conditions: DDA, in which more difficult levels were provided to players based primarily on measures of performance; incremental; and a choice condition, in which players were given the option to select a

## DYNAMIC DIFFICULTY ADJUSTMENT IN VIDEO GAMES

harder or easier level after each completed level. Players in the DDA condition showed significantly higher performance (indexed as reaching more difficult levels more quickly) than players in the other two conditions (see also Baldwin et al., 2014).

However, in a study by Orvis et al. (2008), no significant differences in performance or motivation were found between 4 groups, playing versions of a game with either no difficulty adjustment, incremental difficulty adjustment, or adaptive adjustment (two versions, distinguished in terms of the starting difficulty, which was either easy or hard). Other research on motivation finds similar negative results, with DDA not associated with significantly different levels of player motivation than incremental difficulty adjustment in a Spanish language education game (Sampayo-Vargas et al., 2013). However, in line with previous results showing increased performance, the authors showed that DDA led to significantly higher learning outcomes, which they attribute to a scaffolding effect wherein the reductions in difficulty (the “scaffold”) are provided when students need support, then removed when students were ready to progress.

While Sharek & Wiebe (2015) claim that DDA increases performance with no decrease in engagement, Altimira et al. (2017) found that DDA increased player engagement in a digitally augmented game of table tennis. By projecting images onto a surface, they could increase or decrease the difficulty of the game (e.g., by altering the size of the virtual table projected onto the surface). They found that adjusting the difficulty in response to the score differentials between the two players (e.g., by making one player’s half of the table smaller, thus making the game harder for the opposing player), was associated with significantly higher player engagement (self-report questionnaire) than no adjustment. Preliminary results from research by Xue et al. (2017) using DDA in a mobile game distributed via the Google Play Store and Apple App Store show that DDA increases player engagement over a longer period (4 months). This study is notable as it measures player engagement objectively, in terms of total time spent playing the game. The authors also note that using DDA had no effect on the amount of revenue generated from in-game transactions.

Altimira et al. (2017) also highlight another potential application of DDA technology, in that they showed that using DDA significantly reduced the score differences between players, thus allowing less skilled players to be more competitive against more skilled players. Other studies have successfully used DDA to reduce skill differentials between players of different abilities (e.g., Jensen & Grønbæk, 2016; Hwang et al., 2017). Gerling et al. (2014) created a rhythm game (i.e., in which players must perform steps in time with music) which could either be controlled by a dance mat (i.e., the player inputs the rhythm with their feet), buttons on a standard video game controller, or a via a wheelchair input (wheelchair movements are captured by a motion sensor camera). Using various techniques, they produced adaptive versions of the game which allowed less-skilled able-bodied players to compete with more-skilled able-bodied players, and players with and without mobility disabilities to compete together. Bateman et al. (2011) also decreased performance differentials between players by providing adaptive targeting assistance in a simple shooting game. DDA may therefore be important for enabling people with disabilities to play multiplayer games with those without disabilities (Hernandez et al., 2013; Hwang et al., 2017). DDA may also be one factor which can increase the effectiveness of rehabilitation games for people with disabilities, both in terms of making such games accessible and in terms of adapting the difficulty of the games to suit players of a wide range of abilities and

provide an optimum level of challenge, which is shown to increase the effectiveness of such games (Barrett et al., 2016; Hocine et al., 2015).

To summarise, previous research generally supports the idea that performance-based DDA can increase player performance, and that it can be used to reduce performance differentials between players of different abilities. Proposed applications of this include making games more accessible or inclusive for people with disabilities, and, related to this, increasing the effectiveness of games for rehabilitation. However, in terms of player experience – i.e., engagement, enjoyment, and motivation – previous research provides mixed results on the effects of DDA.

### 1.3 Flow

Several issues relevant to DDA are captured in the concept of flow, which was first introduced by Csikszentmihalyi (1975). During a flow state, an individual is completely focused on an activity; it is an enjoyable and fulfilling experience (an “optimal” experience), often described colloquially as being in “the Zone” (Chen, 2007). Csikszentmihalyi identifies several characteristics of the flow state, including having clear goals, concentrating on the task at hand, feeling in control, and being engaged in a challenging activity requiring skill. Flow was originally modelled by Csikszentmihalyi as an optimal balance between anxiety and boredom; the zone in which one is challenged enough to not be bored, but skilled enough to not be anxious about one’s performance (Csikszentmihalyi, 1988). Since Csikszentmihalyi’s original work, a large body of research has further investigated and characterised flow, and several instruments have been developed for measuring the subjective experience of flow (Weibel et al., 2008). Some of this work has explicitly focused on video games, which have been claimed to “possess ideal characteristics to create and maintain flow experiences in that the flow experience of video games is brought on when the skills of the player match the difficulty of the game” (Sherry, 2004). The importance of the flow experience is shown by empirical research suggesting that flow is one of the reasons why people play video games (Perttula et al., 2017). In educational games, flow has been used as a measure of game quality, and a small amount of research suggests that the experience of flow is associated with the effectiveness of game-based learning (Perttula et al., 2017).

The notion of a balance between player skill and game difficulty shows the direct relevance of flow to performance-based DDA. If the aim of these approaches to DDA is to match the difficulty of a game to each unique player’s skill level, then it seems likely that DDA could be used to achieve a balance between these two factors, and therefore encourage flow experiences. However, although theoretical discussions of flow feature in much research on DDA, we are aware of only one previous study investigating the relationship between difficulty adjustment and flow empirically. Using a modified version of *Tetris* and a within-subjects design with 30 participants, Ang & Mitchell (2017) showed that playing with incremental difficulty adjustment and with a version of DDA in which the player could control the difficulty were both associated with significantly different scores (compared to no DDA) on several constructs of the Flow State Scale (Jackson & Marsh, 1996). This included challenge-skill balance, which was greater when participants played a game with DDA. Note, however, that the DDA used in this study is not performance-based as discussed in this paper, in that players manually (and voluntarily) increased or decreased the difficulty

## DYNAMIC DIFFICULTY ADJUSTMENT IN VIDEO GAMES

during the game themselves, rather than have the difficulty automatically adjusted based on a measurement of player performance.

### 1.4 Machine learning for a generalised adaption mechanism

One of the limitations of many previous DDA systems is their specificity. Much of the research discussed in this paper makes use of context-specific algorithms; that is, the algorithm is designed and implemented for the game at hand, based on “domain-specific requirements” (Chang, 2013). In other words, many previous systems rely on a heuristic that only applies to either a specific game, or a specific genre of games. This limits the extent to which these systems can be readily applied to other games, and limits the applicability of research findings using these systems.

The machine learning approach to DDA developed by Missura & Gärtner (2011) is intended to address this issue. Their partially ordered set master (POSM) algorithm is a generalised DDA mechanism which does not rely on a domain-specific heuristic. A detailed description of the algorithm is beyond the scope of this paper, but the procedure can be summarised as follows. Given a set of possible difficulty settings, some will be too hard for the player and some will be too easy. Furthermore, because difficulty settings can be ordered (i.e., they can be sorted from least to most difficult), it follows that if a setting is judged to be too easy (or too hard), then all settings easier (or harder) than it will also be too easy (or too hard).

Based on this, the POSM algorithm operates by allowing the player to play one round of the game (this can be a level, or a period of any duration); next, based on feedback from the game (expressed simply as “too hard” or “too easy”), the algorithm updates its “belief” (a numerical value) about the suitability of that setting, and all settings easier (or harder) than that setting. The algorithm then selects the setting which currently has the highest belief value as the most appropriate for the player in the next round (see Missura & Gärtner, 2011, for a detailed description of the POSM algorithm). The advantage of this approach is that the algorithm is blind to both what the settings are (they could be game speed, number of enemies, or anything that can be quantified) and the heuristic that determines the feedback; in other words, the message “too hard”/“too easy” is sent from the game to the POSM algorithm, and deciding which message to send is the responsibility of the game, not the POSM algorithm. This shows how POSM is generalised: it selects settings based on feedback, but it does not know what those settings are or how the feedback is determined. As such, it should be possible to easily apply POSM in almost any game, without modification. However, as far as we are aware, POSM has only been implemented and evaluated (with human players, as opposed to simulations) in turn-based strategy games (Illici et al., 2012). Therefore, the feasibility of such an approach to DDA in other kinds of video game remains to be investigated.

### 1.5 The present study

Our aim in this study was to investigate how a performance-based approach to DDA based on POSM affects player performance, enjoyment, and experience of flow, using an experimental game created for this study. We conducted a controlled experiment with 3 groups, with each group playing a different version of the game. The independent variable was the way in which the difficulty of the game was adjusted, with 3 levels: (1) DDA, (2) incremental difficulty adjustment (in which the game gets progressively harder irrespective of player experience), and (3) no difficulty adjustment (control group). In the DDA version of the game, we used a generalised machine learning algorithm (based on POSM) to adjust the

## DYNAMIC DIFFICULTY ADJUSTMENT IN VIDEO GAMES

difficulty, and a measure of player performance provided the feedback upon which the adjustments were based.

### 1.5.1 *Hypotheses*

Our hypotheses were (1) DDA will produce greater player performance than either incremental or no difficulty adjustment; (2) DDA will lead to greater experience of flow than either incremental or no difficulty adjustment; and (3) DDA will lead to greater enjoyment than either incremental or no difficulty adjustment.

## 2 *Materials and methods*

### 2.1 *The game*

To test our hypotheses, we designed and implemented a simple video game called “Meteor Shower” (Figure 1). The object of the game is to avoid the meteors which continually fall from the top of the screen while catching the pink falling stars. The player controls the yellow character by moving left or right along the bottom of the screen (using the left and right directional arrows on the keyboard). The velocity of the meteors determines the difficulty of the game, with higher velocities making the game more difficult (i.e., the meteors are harder to avoid). The game consists of 20 levels, each lasting 45 seconds, with a short pause between each level. Players are awarded one point for each star they catch, and lose a point each time a meteor hits the character. The score is reset to 0 at the end of each level. There are 8 falling stars to catch in each level (each falling 5 seconds apart), and so the maximum score available on any level is 8 (i.e., the player catches all stars and avoids all meteors). While the meteors and stars appear to the player to originate from random locations, the game in fact uses a seeded random number generator to ensure that the pattern of locations at which stars and meteors appear is the same for each player. When generated, stars fall in a straight line. Meteors move in a straight line from their point of origin, to the location of the player-controlled character at the time the meteor was generated. This ensures that players always must move the character to avoid every meteor. Three versions of the game were created. The sole difference between each version is the way in which the difficulty (i.e., the velocity of the meteors) was adapted during gameplay.

## DYNAMIC DIFFICULTY ADJUSTMENT IN VIDEO GAMES

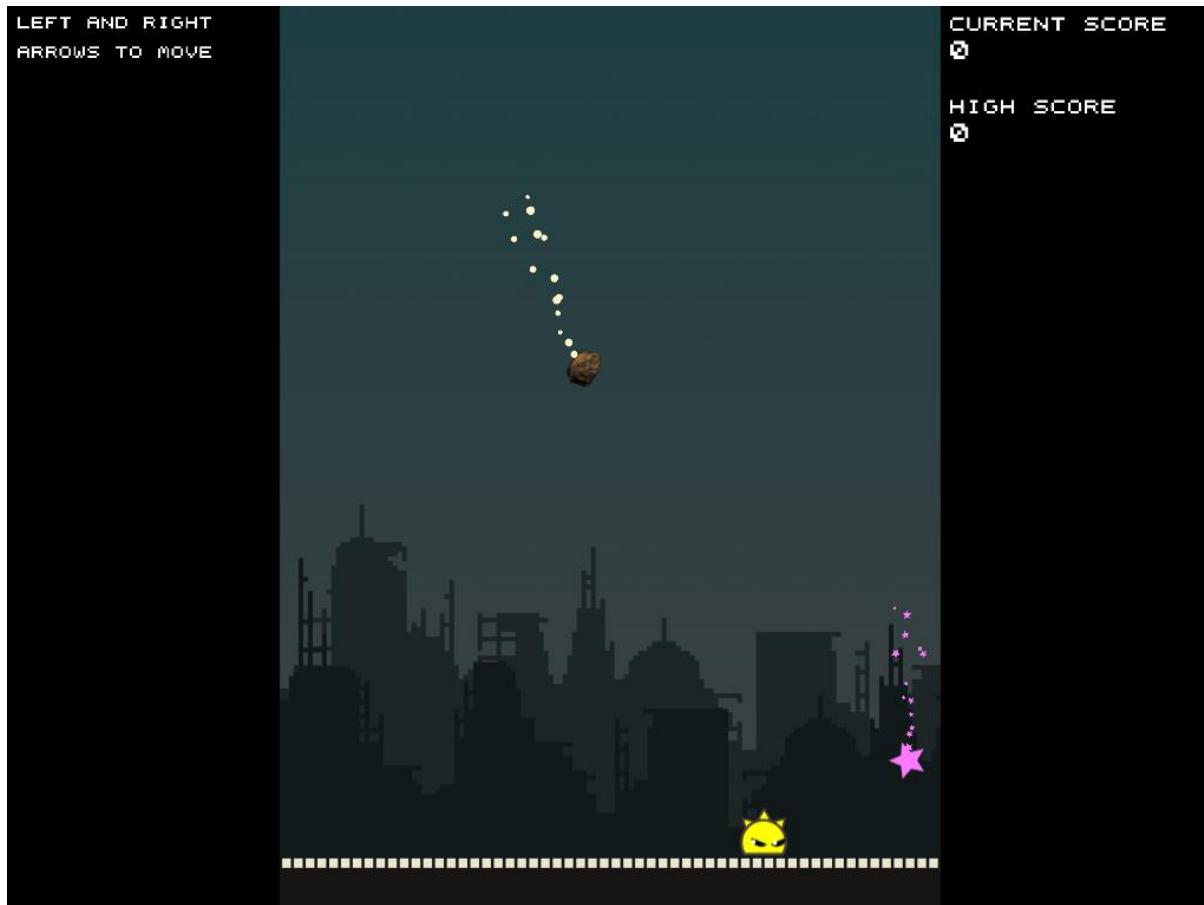


Figure 1. The experimental game, Meteor Shower. The player moves the yellow character left and right using the directional arrows on a keyboard. The meteor, which is moving towards the yellow character, should be avoided. The pink star, which is falling in a straight line, should be collected. The velocity at which the meteor falls dictates the difficulty of the game (faster is more difficult).

### 2.1.1 Control version of the game

In this version of the game, the velocity of the meteors remained constant at 800 throughout.

### 2.1.2 Incremental version of the game

In this version of the game, the velocity of the meteors increased by 50 at the start of each new level. The velocity on level 1 was 200; the velocity on level 20 was 1150.

### 2.1.3 DDA version of the game

This version of the game used a machine learning algorithm, based on the POSM algorithm, to adapt the velocity of the meteors in response to measurements of player performance. The starting value on level 1 was 900. The possible settings ranged from 200 to 1700, increasing in increments of 50. Due to an error in the programming, value 1200 was not used.

As described in Section 1.4, the POSM algorithm requires a message to be regularly sent from the game describing if the current setting is too hard or too easy for the player. In this study, this message was sent every 5 seconds. The decision was based on two measures of player success rate. The first of these measured the player's success at avoiding meteors over the previous 5 seconds of play ( $success1 = (nM - nH)/nM$ , where  $nM$  is the number of meteors in the previous 5 seconds and  $nH$  is the number of times the player was hit by a

meteor in the previous 5 seconds). The second measurement was the player's overall success rate based on their score and the potential maximum score possible at that point in the level ( $\text{success2} = \text{score}/nS$ , where  $nS$  is the number of stars that have fallen so far in the level).

Determining which value to use for the target success rate (i.e., the value above which the game was judged to be too easy), proved to be one of the most challenging design decisions in the development of the game, and we were unable to find previous research to guide this decision. Therefore, during the development process, we played versions of the game using success rates ranging from 0.75 (i.e., 75%) to 1 (i.e., 100%). We determined that the most satisfying experience was provided when we used a target success rate of 1 for both measures.

## 2.2 Questionnaire

Flow was assessed using the Flow Short Scale (Rheinberg et al., 2003), which is shown to be a reliable measure of flow experience in video games (Wiebel et al., 2008). We used the online version of the scale (<http://www.psych.uni-potsdam.de/people/rheinberg/messverfahren/fks1-e.html>) which has 14 items. Items 1 – 10 measure flow, items 11 – 13 measure anxiety, and item 14 measures perceived skill demands (challenge) (see Appendix in Engeser, 2012). In addition, demographic information (age, gender, frequency of video game play) was collected.

## 2.3 Performance data

During gameplay, each participant's score was recorded for levels 1 – 19. Due to a bug in the software (which we identified after running the experiment), the score for level 20 was not recorded. For the DDA group only, we also recorded the velocity of the meteors at 5 second intervals (i.e., each time the velocity was updated). This provided a measure of the difficulty of the game (higher velocity is more difficult), and a measure of the player's skill level (better players will reach higher velocities). Scores were recorded for all participants in the DDA group for 855 seconds of gameplay (i.e., not the full 15 minutes, due to a technical problem or bug, currently unidentified). We did not record velocity for the control group, as this remained constant throughout the game (800) or for the incremental group, as this increased on a pre-determined scale with each level.

## 2.4 Data collection method: Amazon Mechanical Turk

We conducted the experiment online using Amazon Mechanical Turk (MTurk). MTurk has been described as a “marketplace for work that requires human intelligence” (Rouse, 2015). Users of the site are classified as “requesters” (who post tasks to the site) and “workers” (who complete these tasks in return for payment). Example tasks include completing surveys to provide feedback about a website, classifying images based on their content, or providing translations of short pieces of text. Typically, MTurk is appropriate for tasks which can be completed quickly, and for which many instances of the task must be completed. MTurk is now frequently used to conduct research in psychology (Paolacci & Chandler, 2014; Mason & Suri, 2012), and it has been used in at least one previous study on the effects of DDA in video games (Sharek & Wiebe, 2015).

However, several issues have been identified which may threaten the validity of data obtained from MTurk (Cheung et al., 2017). Some of these issues are not unique to MTurk. For example, the issue of selection bias, in the sense that MTurk workers choose the tasks

they wish to complete, applies in any research wherein participants voluntarily opt to take part after viewing, e.g., a poster or advertisement on social media. However, all participants in research conducted via MTurk will (obviously) have self-selected to become MTurk workers. Related to this, Cheung et al. (2017) note that samples obtained from MTurk may not be representative of the population of interest (e.g., all MTurk participants are internet users, which not be appropriate for some studies). On this issue, we make two observations. Firstly, we note that, in some respects, samples obtained from MTurk may be more diverse than samples obtained by traditional means. Casler et al. (2013) point out that most participants in psychological research are American college students; they showed that a sample of MTurk workers was significantly, desirably more diverse in terms of ethnicity, economic status, and age, than a sample of undergraduate students. Secondly, we point out that the nature of our study – in which participants are required to play an online video game remotely – dictates that participants would be required to have internet access and be reasonably computer literate whether recruited through MTurk or not.

Perhaps the most important concern with MTurk data highlighted by Cheung et al. (2017) is the possibility of participants answering questions without paying attention to the content (either fully, or at all, i.e., selecting random answers). However, steps can be taken to mitigate this risk (see also Fleischer et al., 2015). Firstly, MTurk incorporates a rating system for workers, so that requesters can specify that only workers of a suitable quality can access their tasks. We will discuss this further in Section 2.6, where we describe how we used this system to specify that only workers of a certain quality could access our experiment. Secondly, items can be included in questionnaires to check for attentiveness (e.g., a multiple-choice item that states which option the respondent should select). Finally, it may also be possible to detect inattentive responses by analysing data gathered, although this would presumably depend on the nature of the data. The screening process we used to identify inattentive participants in the present study is described in Section 2.7.

As should be the case in all data collection processes, we recommend that consideration is given to potential limitations of data collected using MTurk, and that these limitations are addressed as fully as possible.

## 2.5 Procedure

The task was made available using MTurk's Survey Link template. This provides a link to an external website, where participants complete a task (typically a questionnaire) and receive a completion code. They then enter the completion code in MTurk to receive credit for completing the task. The default configuration of the Survey Link template allows each worker to only complete the task once. In our case, the survey link took participants to a site hosting the game. Which version of the game was loaded was determined randomly by the software. Participants pressed a button to start the game when they were ready. After 15 minutes of play, the game automatically ended, and participants were presented with a link to the webpage containing the questionnaire. When they submitted the questionnaire with all questions completed, the data were stored on a server, and a unique completion code was generated on the server and returned to participants. The completion code, a record of which version of the game they had played, and performance data automatically recorded during gameplay, were also stored on the server. Participants then entered the completion code in MTurk, and were paid \$0.99. All participants were paid, whether their data were included in the analysis or not.

## 2.6 Pilot

Initially, we ran a pilot with 10 participants. By considering participants' performance data, it appeared that some participants did not actually play the game. It would be possible to merely let the game run for 15 minutes, then select random answers to the questions. To address this, we included two attention check items in the questionnaire. These items stated that the participant should select option 7 ("very much") in the Likert-style scale. We also screened the data collected during the full experiment and removed data which appeared to show no engagement with the game (see Section 2.7). In addition, we decided to use MTurk's qualifications feature to ensure that the task was only available to workers who (1) had completed over 10000 tasks on MTurk; and (2) and an approval rate of 97% or higher.

## 2.7 Participants

Data were collected from 300 adults via MTurk, each randomly assigned to play either the control, DDA, or incremental versions of the game. Entries in which answers to either of the attention check questions were incorrect were removed. We also removed entries in which a score of zero was recorded for every level, as this suggested that the participants did not actually play the game, but merely let it run for the required time. Finally, we considered the velocity data recorded for participants in the DDA group, and removed any entries in which the pattern of velocities across the levels suggested that participants had not actually played the game (i.e., when the velocity quickly decreased to 200 and did not rise above 250 for the remainder of the game).

This left 221 participants (93 female) whose data were retained for analysis. Ages ranged from 20 years to 65 years, with a mean age of 36.51 years (std. deviation 9.73). There were 82 participants in the control group, 68 in the DDA group, and 71 in the incremental group.

## 3 Results

The 14-item online version of the Flow Short Scale was shown to have acceptable reliability (Cronbach's  $\alpha = .834$ ).

Chi-square tests of homogeneity showed that the three groups did not significantly differ in terms of gender ( $\chi^2(3) = 1.4$ ,  $p = .497$ ), number of days per week spent playing video games ( $\chi^2(3) = 23.348$ ,  $p = .055$ ), or age ( $\chi^2(3) = 69.525$ ,  $p = .902$ ).

### 3.1 Significant results: performance

To analyse group differences in terms of overall mean score (i.e., participants' mean score over 19 levels of play), we ran a Kruskal-Wallis H test. The distributions of overall mean score were not similar for all groups. The DDA group had a smaller range (2.89) and smaller interquartile range (.83) than both the control group (range = 7.32, interquartile range = 3.01) and incremental group (range = 7.74, interquartile range = 2.89) (see Figure 2). The median values increased from the incremental group (3.74), to the control group (4.61) to the DDA group (5.05). These differences in median values were statistically significantly different between the groups,  $\chi^2(3) = 16.148$ ,  $p < .0005$ . Pairwise comparisons were then performed using Dunn's (1964) procedure with a Bonferroni correction for multiple comparisons. This analysis revealed statistically significant differences (adjusted p-values presented) in median values between the DDA group (mean rank = 135.31) and control

## DYNAMIC DIFFICULTY ADJUSTMENT IN VIDEO GAMES

group (mean rank = 106.91) ( $p = .02$ ), and between the DDA group and incremental group (mean rank = 92.44) ( $p < .0005$ ) groups, but not between the control group and the incremental group ( $p = .488$ ).

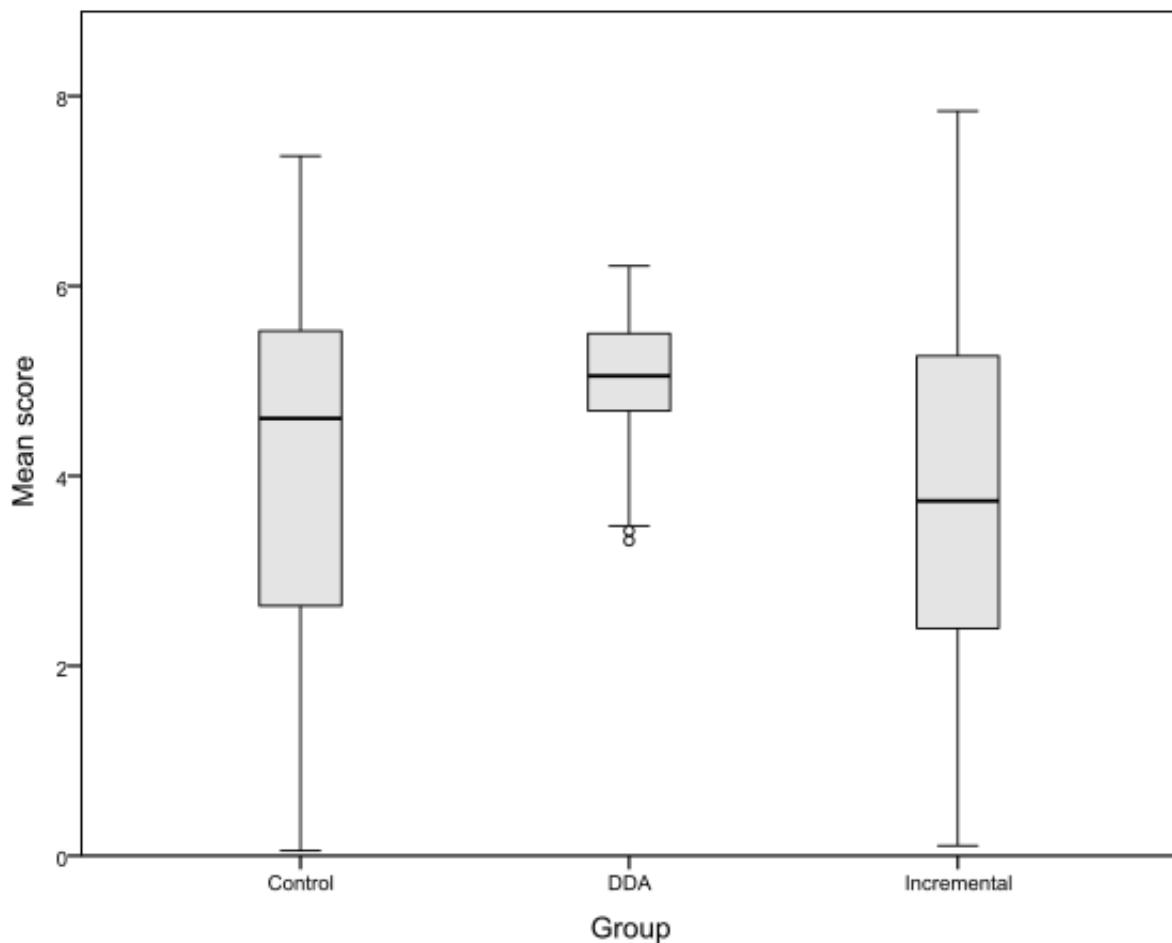


Figure 2. Boxplots of overall score for each of three groups, Control, Dynamic Difficulty Adjustment (DDA), and Incremental.

We analysed the differences in scores between the three groups further by conducting one-way Welch ANOVAs on mean score per level (i.e., mean score for each group for each of 19 levels of play). A small number of outliers in the DDA group and Incremental group were not removed. Levene's test for equality of variances showed that the assumption of homogeneity of variances was violated in levels 4 – 19 (p-values ranging from .012 to <.0005). Scores were significantly different between the groups during levels 1, 3, 5, and levels 9 – 19. Games-Howell post hoc analyses revealed that the DDA group had higher mean scores than the control group in levels 3 – 19; these differences were significant in levels 3, 5, and 9 – 19 (p-values ranging from .019 to <.0005). The DDA group also had significantly higher mean scores than the incremental group in levels 10 – 19 (p-values ranging from .013 to <.0005). The control group had significantly higher mean scores than the incremental group in levels 14 – 19 (p-values ranging from .014 to <.0005). These results are shown in Figures 3.1, 3.2, and 3.3.

## DYNAMIC DIFFICULTY ADJUSTMENT IN VIDEO GAMES

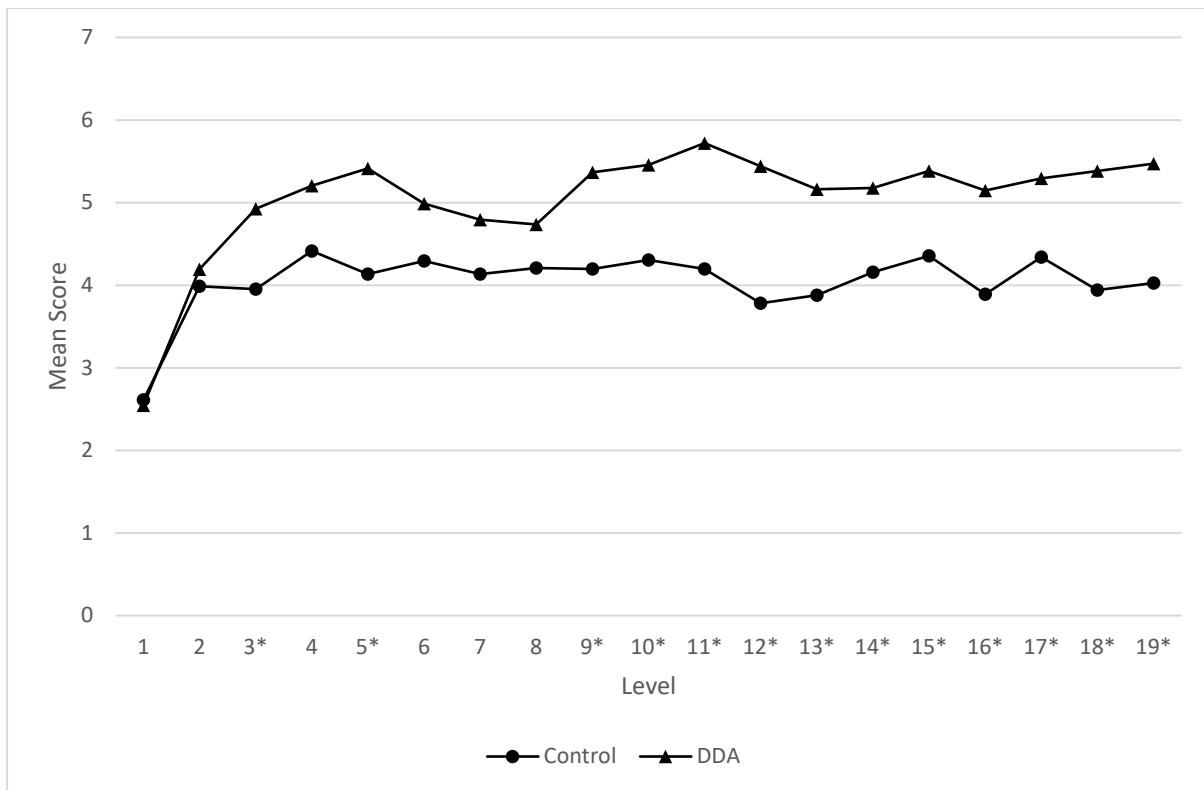


Figure 3.1. A comparison of mean score per level for the Control and Dynamic Difficulty Adjustment (DDA) groups. Significant differences are marked with a \*.

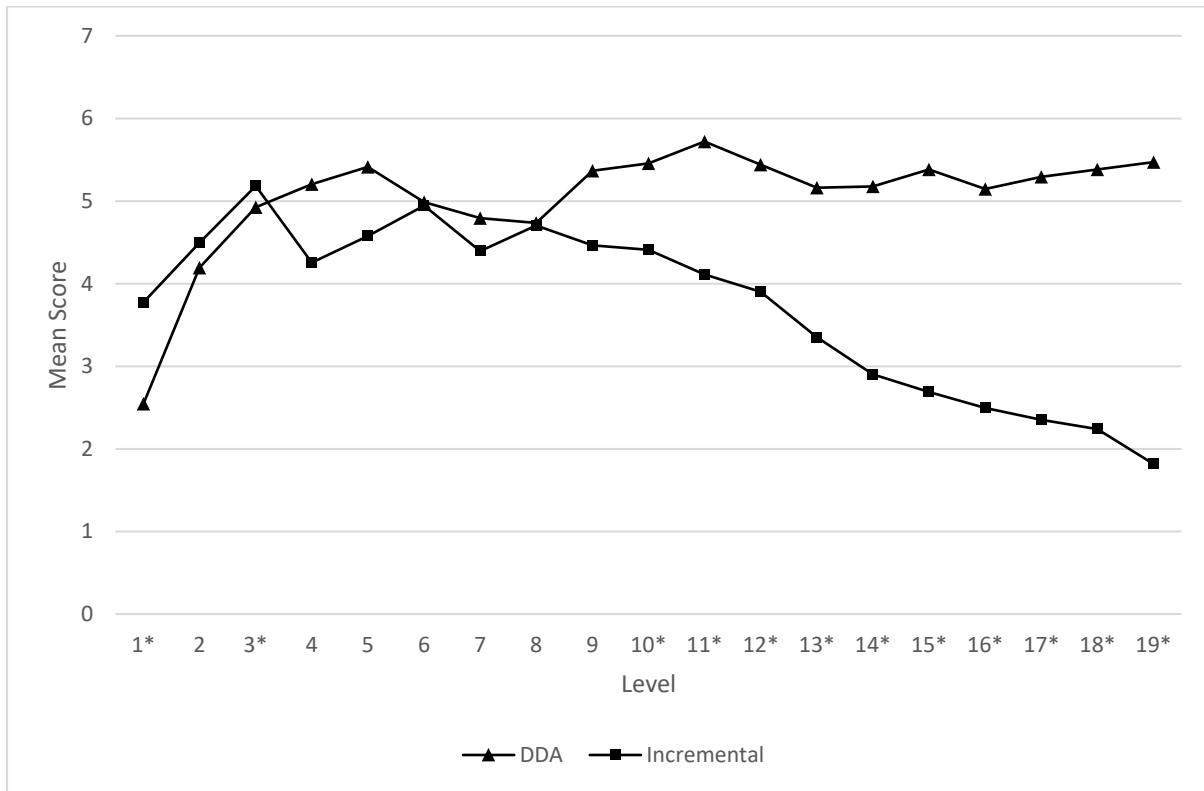


Figure 3.2. A comparison of mean score per level for the Dynamic Difficulty Adjustment (DDA) and Incremental groups. Significant differences are marked with a \*.

## DYNAMIC DIFFICULTY ADJUSTMENT IN VIDEO GAMES

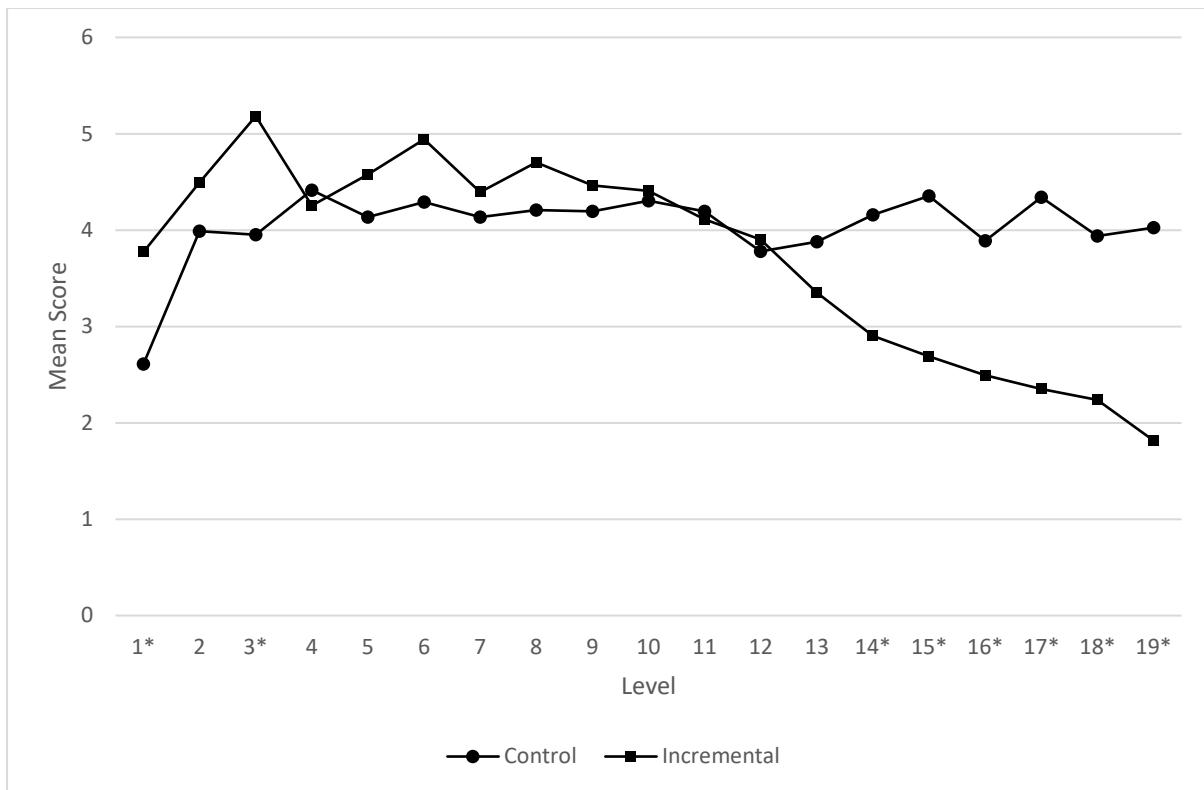


Figure 3.3. A comparison of mean score per level for the Control and Incremental groups. Significant differences are marked with a \*.

For the DDA group, we analysed the velocity values, which indicates the difficulty of the game (higher velocity is more difficult). First, we considered the mean velocity across participants at each measurement point (5 seconds between each measurement, 171 measurements considered). Figure 4 shows how velocity ranged across participants over time. We considered the relationship between each participant's (DDA group only) overall mean velocity over 855 seconds of gameplay, and each participant's overall mean score across 19 levels, using Pearson's correlation test. The data were linear, overall mean score was normally distributed ( $p > .05$ ), while overall mean velocity was not normally distributed ( $p < .0005$ ). There was a strong positive correlation between overall mean score and overall velocity,  $r(68) = .554$ ,  $p < .0005$ , with overall mean velocity explaining 31% of the variation in overall mean score

## DYNAMIC DIFFICULTY ADJUSTMENT IN VIDEO GAMES

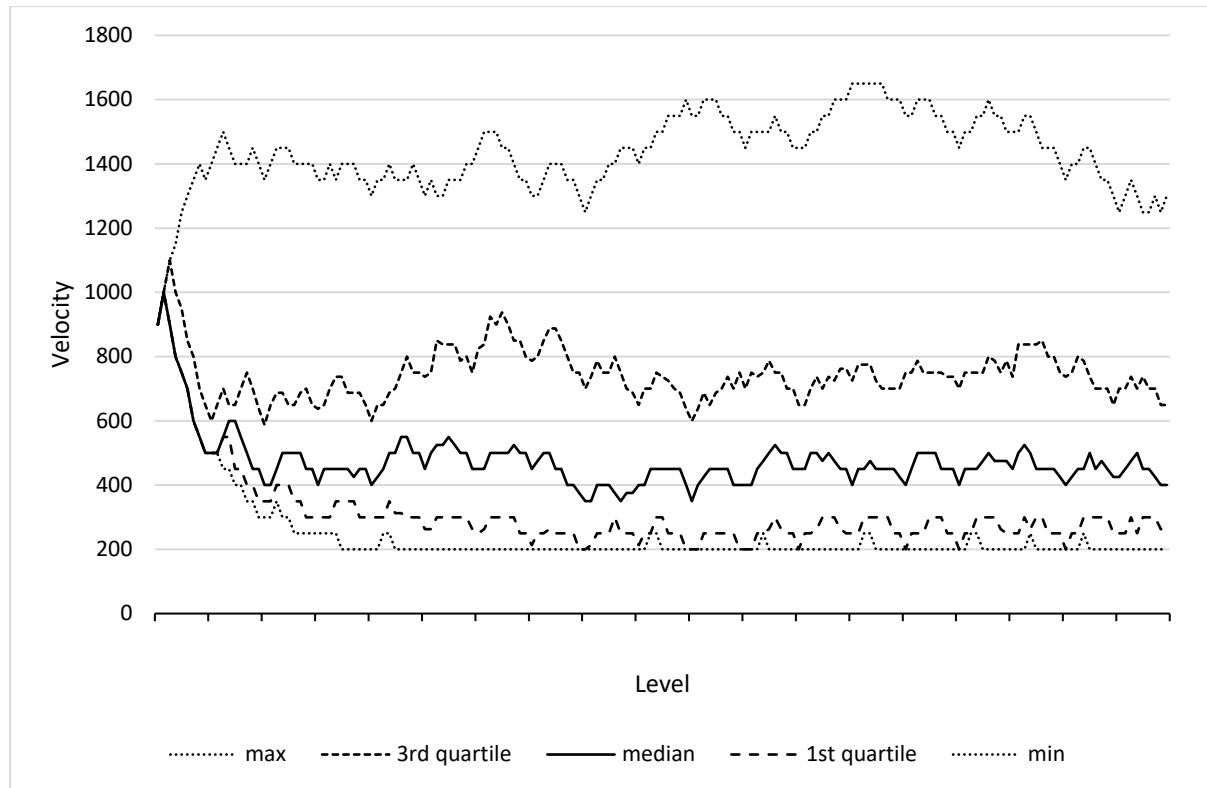


Figure 4. Range of meteor velocity per level for the Dynamic Difficulty Adjustment (DDA) group over 19 levels. As a higher velocity makes the game more difficult, and higher velocities are only achieved by players who perform better, velocity here can be used as an index of player performance.

### 3.2 Non-significant results: flow, anxiety, challenge, and enjoyment

We also ran Kruskal-Wallis H tests to analyse differences between the groups in terms of the 3 factors of the Flow Short Scale, and the single-item for Enjoyment.

In the case of Flow, the distributions were not similar for all groups (assessed by visual inspection of a box plot). The median values increased from the incremental group (4.9), to the DDA group (4.95), to the control group (5.3). These differences were not statistically significantly different between the groups,  $\chi^2(3) = 4.087$ ,  $p = .130$ .

In the case of Anxiety, the distributions were similar for all groups. The median values increased from the incremental group (3.67), to the control group (4.0) and DDA group (4.0). These differences were not statistically significantly different,  $\chi^2(3) = 3.336$ ,  $p = .189$ .

In the case of Challenge, the distributions were not similar for all groups and the median values were the same for all three groups (4.0).

In the case of Enjoyment, the distributions were not similar for all groups. The median values increased from the incremental group (5.0), to the control group (6.0) and the DDA group (6.0). These differences were not statistically significantly different,  $\chi^2(3) = 3.628$ ,  $p = .163$ .

### 3.3 Correlations

We also considered relationships between flow, anxiety, challenge, enjoyment, and overall mean score across all three groups. Flow was positively correlated with Anxiety ( $r(221) = .462$ ,  $p < .0005$ ), Challenge ( $r(221) = .476$ ,  $p < .0005$ ), and Enjoyment ( $r(221) = .675$ ,  $p < .0005$ ), Anxiety was positively correlated with Challenge ( $r(221) = .531$ ,  $p < .0005$ ) and

Enjoyment ( $r(221) = .511, p < .0005$ ), and Challenge was positively correlated with Enjoyment ( $r(221) = .501, p < .0005$ ). Overall mean score was significantly negatively correlated with Challenge ( $r(221) = -.181, p = .007$ ).

### 3.4 Hypotheses

In the case of hypothesis (1) – that DDA will produce greater player performance than either incremental or no difficulty adjustment – we are able to reject the null hypothesis.

However, for hypotheses (2) and (3), we are unable to reject the nulls. That is, we cannot reject the hypotheses that there is no difference between DDA, incremental difficulty adjustment, and the control group in terms of either player experience of flow or player enjoyment.

## 4 Discussion, limitations, and future work

The research presented here investigated dynamic difficulty adjustment in a video game, with the adjustment based on feedback about the player's performance, and using a machine learning algorithm to select the appropriate setting for a single variable which directly affected game difficulty. We found that this type of difficulty adjustment led to greater overall player performance over approximately 15 minutes of play, than either incremental difficulty adjustment or a no-adjustment control group. In addition, the range of performance in the DDA group was smaller than the other groups, and overall performance in the DDA group correlated with overall mean difficulty across 15 minutes of play.

In line with previous research, our results suggest that performance-based DDA is suitable for reducing skill differentials between players. This provides further evidence of the suitability of this relatively simple approach to DDA in facilitating competitive and/or collaborative play between players with different skill levels. We believe that this approach could therefore be used to increase the accessibility of video games for people with disabilities.

It is also interesting to note that the players in the DDA group showed a wide range of abilities, as indexed by the range of difficulty settings recorded during the experiment (Figure 4). However, the overall mean performance of this group increased more than both the control and incremental groups, with significant differences in mean performance in most levels of the game between the DDA and other groups. It is therefore possible, in line with the results of Sampayo-Vargas et al. (2013), that DDA provides a scaffold to players of a range of abilities, by making the game easier when their performance drops; this scaffold is then removed when their performance increases. However, there may be detrimental effects associated with DDA if players are aware that it is operating (Baldwin et al., 2014; Baldwin et al., 2016), and further research should investigate how players' awareness of DDA is related to their experience of playing a video game.

Our results show no significant differences between the three conditions on all the self-reported measures of player experience (flow, enjoyment, challenge, and anxiety). Previous research on performance-based DDA has found mixed results on self-report measures of player experience such as enjoyment, engagement, motivation, and challenge. There are several possible explanations for this. Firstly, it may be the case that performance-based approaches to DDA have less or no effect on self-reported player experience than affective

approaches. This is feasible, as several studies have found that player perceptions of gameplay experience are related to factors other than player skill or performance, such as gameplay experience (Alexander et al., 2013), motivations, personality, or preferences (Karpinskyj et al., 2014). If the aim of DDA is to increase players' positive perception of the gameplay experience, then affective approaches to DDA may be more useful. Secondly, within performance-based DDA approaches, there is a large range of factors which could influence player experience. In this study, we used simple measures of player performance to alter a single variable affecting game difficulty. There are many other factors we could have chosen. In addition, we could have provided a different range of difficulty settings (e.g., with larger or smaller increments), used a different target success rate, adapted the difficulty more or less frequently, and so on. These adjustments could lead to different results, and future research should consider, not just the difference between adaptive and non-adaptive difficulty adjustment, but also differences between alternative approaches to DDA. Thirdly, as discussed in Section 1.4, self-report data obtained from MTurk may be less reliable than data obtained from traditional sources. While we included attention-check items to identify participants who were potentially selecting random answers to the questions, found high reliability for our questionnaire, and found expected correlations between flow, anxiety, challenge, and enjoyment, we did not use any other techniques to identify participants who were not engaged with the questionnaire. For example, it has been suggested that MTurk data can be made more reliable by explicitly asking participants if they answered the questions genuinely and assuring them that they will still be paid if they admit they did not (Rouse, 2015). Note that this limitation is somewhat mitigated in this study as we removed responses from participants whose performance data indicated that they had not engaged with the game. However, it is still feasible that some participants engaged with the game and read the questions but still provided unreliable data simply by not providing considered answers.

While this study focused on a performance-based approach to DDA, it is important to recall that the POSM algorithm used is not specific to this context or to the game used in our experiment. This means that our approach, in principle, could be matched with any other feedback that can be quantified. It could therefore be used, not only with other performance measures, but, crucially, with affective measures, as any real-time quantitative data could be used as the feedback upon which difficulty is updated. This could include, for example, heart rate, galvanic skin response, or neurological activity. In future work, it would be informative to test the feasibility of the POSM algorithm in affective DDA systems.

## 5 Conclusion

This study makes several contributions to research on dynamic difficulty adjustment in video games. Our results show that performance-based dynamic difficulty adjustment, based on the Partially ordered set master (POSM) algorithm (Missura & Gärtner, 2011) can be used to increase player performance and reduce performance differentials in a 2D video game. We also demonstrate the feasibility of conducting video games research using an experimental game via Amazon Mechanical Turk. We make recommendations for future research to further investigate the effects of dynamic difficulty adjustment on enjoyment and experience of flow (for which we found non-significant results), such as using different feedback measures (including affective feedback), adjusting different game variables, and

## DYNAMIC DIFFICULTY ADJUSTMENT IN VIDEO GAMES

implementing additional steps to ensure the reliability of data gathered by player self-report.

### References

- Adams, E. (2008, May 14). The designer's notebook: Difficulty modes and dynamic difficulty adjustment. Retrieved from [https://www.gamasutra.com/view/feature/132061/the\\_designers\\_notebook\\_.php](https://www.gamasutra.com/view/feature/132061/the_designers_notebook_.php)
- Alexander, J. T., Sear, J., & Oikonomou, A. (2013). An investigation of the effects of game difficulty on player enjoyment. *Entertainment Computing*, 4(1), 53-62.
- Altimira, D., Clarke, J., Lee, G., Billinghurst, M., & Bartneck, C. (2017). Enhancing player engagement through game balancing in digitally augmented physical games. *International Journal of Human-Computer Studies*, 103, 35-47.
- Ang, D. & Mitchell, A. (2017). Comparing effects of dynamic difficulty adjustment systems on video game experience. In Proceedings of the Annual Symposium on Computer-Human Interaction in Play, 317-327. ACM.
- Baldwin, A., Johnson, D., & Wyeth, P. A. (2014). The effect of multiplayer dynamic difficulty adjustment on the player experience of video games. In *Proceedings of the extended abstracts of the 32nd annual ACM conference on Human factors in computing systems* (pp. 1489-1494). ACM.
- Baldwin, A., Johnson, D., & Wyeth, P. (2016). Crowd-pleaser: Player perspectives of multiplayer dynamic difficulty adjustment in video games. In *Proceedings of the 2016 Annual Symposium on Computer-Human Interaction in Play* (pp. 326-337). ACM.
- Barrett, N., Swain, I., Gatzidis, C., & Mecheraoui, C. (2016). The use and effect of video game design theory in the creation of game-based systems for upper limb stroke rehabilitation. *Journal of Rehabilitation and Assistive Technologies Engineering*, 3, 2055668316643644.
- Bateman, S., Mandryk, R. L., Stach, T., & Gutwin, C. (2011). Target assistance for subtly balancing competitive play. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 2355-2364). ACM.
- Bevilacqua, F., Backlund, P., & Engstrom, H. (2015). Proposal for non-contact analysis of multimodal inputs to measure stress level in serious games. In *Games and Virtual Worlds for Serious Applications (VS-Games), 2015 7th International Conference on*. IEEE.
- Bevilacqua, F., Backlund, P., & Engstrom, H. (2016). Variations of Facial Actions While Playing Games with Inducing Boredom and Stress. In *Games and Virtual Worlds for Serious Applications (VS-Games), 2016 8th International Conference on*. IEEE.
- Booth, M. (2009). The AI systems of Left 4 Dead. Keynote presented at the 5<sup>th</sup> Artificial Intelligence and Interactive Digital Entertainment Conference, Stanford, CA.
- Bontchev, B. (2016). Adaptation in Affective Video Games: A Literature Review. *Cybernetics and Information Technologies*, 16(3), 3-34.
- Casler, K., Bickel, L., & Hackett, E. (2013). Separate but equal? A comparison of participants and data gathered via Amazon's MTurk, social media, and face-to-face behavioral testing. *Computers in Human Behavior*, 29(6), 2156-2160.

## DYNAMIC DIFFICULTY ADJUSTMENT IN VIDEO GAMES

- Chang, D. M. J., (2013). Dynamic difficulty adjustment in computer games. Retrieved from <http://studylib.net/doc/8266212/dynamic-difficulty-adjustment-in-computer-games>
- Chen, J. (2007). Flow in games (and everything else). *Communications of the ACM*, 50(4), 31-34.
- Cheung, J. H., Burns, D. K., Sinclair, R. R., & Sliter, M. (2017). Amazon Mechanical Turk in organizational psychology: An evaluation and practical recommendations. *Journal of Business and Psychology*, 32(4), 347-361.
- Csikszentmihalyi, M. (1975). Beyond boredom and anxiety: Experiencing flow in work and play. San Francisco: Jossey-Bass.
- Csikszentmihalyi, M. (1988). The flow experience and its significance for human psychology. In M. Csikszentmihalyi & I. Csikszentmihalyi (Eds.), *Optimal experience: Psychological studies of flow in consciousness* (pp. 15–35). Cambridge: Cambridge University Press.
- Dunn, O. J. (1964). Multiple comparisons using rank sums. *Technometrics*, 6(3), 241-252.
- Engeser, S. (Ed.). (2012). *Advances in flow research*. Springer Science & Business Media.
- Entertainment Software Association (2017). Essential facts about the computer and video game industry. [http://www.theesa.com/wp-content/uploads/2017/09/EF2017\\_Design\\_FinalDigital.pdf](http://www.theesa.com/wp-content/uploads/2017/09/EF2017_Design_FinalDigital.pdf)
- Fleischer, A., Mead, A. D., & Huang, J. (2015). Inattentive responding in MTurk and other online samples. *Industrial and Organizational Psychology*, 8(2), 196-202.
- Gerling, K. M., Miller, M., Mandryk, R. L., Birk, M. V., & Smeddinck, J. D. (2014). Effects of balancing for physical abilities on player performance, experience and self-esteem in exergames. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems* (pp. 2201-2210). ACM.
- Hernandez, H. A., Ye, Z., Graham, T. C., Fehlings, D., & Switzer, L. (2013). Designing action-based exergames for children with cerebral palsy. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 1261-1270). ACM.
- Hocine, N., Gouaïch, A., Cerri, S. A., Mottet, D., Froger, J., & Laffont, I. (2015). Adaptation in serious games for upper-limb rehabilitation: an approach to improve training outcomes. *User Modeling and User-Adapted Interaction*, 25(1), 65-98.
- Hunicke, R. (2005). The case for dynamic difficulty adjustment in games. In *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology* (pp. 429-433). ACM.
- Hwang, S., Schneider, A. L. J., Clarke, D., Macintosh, A., Switzer, L., Fehlings, D., & Graham, T. C. (2017). How Game Balancing Affects Play: Player Adaptation in an Exergame for Children with Cerebral Palsy. In *Proceedings of the 2017 Conference on Designing Interactive Systems* (pp. 699-710). ACM.
- Ilici, L., Wang, J., Missura, O., & Gärtner, T. (2012). Dynamic difficulty for checkers and Chinese chess. In *Computational Intelligence and Games (CIG), 2012 IEEE Conference on* (pp. 55-62). IEEE.

## DYNAMIC DIFFICULTY ADJUSTMENT IN VIDEO GAMES

- Jackson, S. A., & Marsh, H. W. (1996). Development and validation of a scale to measure optimal experience: The Flow State Scale. *Journal of sport and exercise psychology, 18*(1), 17-35.
- Jensen, M. M., & Grønbæk, K. (2016). Design strategies for balancing exertion games: A study of three approaches. In *Proceedings of the 2016 ACM Conference on Designing Interactive Systems* (pp. 936-946). ACM.
- Karpinskyj, S., Zambetta, F., & Cavedon, L. (2014). Video game personalisation techniques: A comprehensive survey. *Entertainment Computing, 5*(4), 211-218.
- Leiker, A. M., Bruzi, A. T., Miller, M. W., Nelson, M., Wegman, R., & Lohse, K. R. (2016). The effects of autonomous difficulty selection on engagement, motivation, and learning in a motion-controlled video game task. *Human movement science, 49*, 326-335.
- Mason, W., & Suri, S. (2012). Conducting behavioral research on Amazon's Mechanical Turk. *Behavior research methods, 44*(1), 1-23.
- Missura, O., & Gärtner, T. (2011). Predicting dynamic difficulty. In *Advances in Neural Information Processing Systems* (pp. 2007-2015).
- Nagle, A., Novak, D., Wolf, P., & Riener, R. (2014). The effect of different difficulty adaptation strategies on enjoyment and performance in a serious game for memory training. In *Serious Games and Applications for Health (SeGAH), 2014 IEEE 3rd International Conference on* (pp. 1-8). IEEE.
- Orvis, K. A., Horn, D. B., & Belanich, J. (2008). The roles of task difficulty and prior videogame experience on performance and motivation in instructional videogames. *Computers in Human behavior, 24*(5), 2415-2433.
- Paolacci, G., & Chandler, J. (2014). Inside the Turk: Understanding Mechanical Turk as a participant pool. *Current Directions in Psychological Science, 23*(3), 184-188.
- Perttula, A., Kiili, K., Lindstedt, A., & Tuomi, P. (2017). Flow experience in game based learning—a systematic literature review. *International Journal of Serious Games, 4*(1).
- Rheinberg, F., Vollmeyer, R., Engeser, S. (2003). Die Erfassung des Flow-Erlebens [Measuring flow experiences]. In J. Stiensmeier-Pelster, F. Rheinberg (Eds.), *Diagnostik von Motivation und Selbstkonzept. Tests und Trends, Vol. 2*, Hogrefe, Göttingen (2003), pp. 261-279
- Robb, N., Waller, A., & Woodcock, K. A. (2019). Developing a task switching training game for children with a rare genetic syndrome linked to intellectual disability. *Simulation & Gaming, 50*(2), 160-179.
- Rouse, S. V. (2015). A reliability analysis of Mechanical Turk data. *Computers in Human Behavior, 43*, 304-307.
- Sampayo-Vargas, S., Cope, C. J., He, Z., & Byrne, G. J. (2013). The effectiveness of adaptive difficulty adjustments on students' motivation and learning in an educational computer game. *Computers & Education, 69*, 452-462. doi: 10.1016/j.compedu.2013.07.004
- Shaker, N., Yannakakis, G., & Togelius, J. (2010). Towards automatic personalized content generation for platform games. In 6th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, AIIDE 2010.

## DYNAMIC DIFFICULTY ADJUSTMENT IN VIDEO GAMES

- Sharek, D., & Wiebe, E. (2015). Investigating Real-time Predictors of Engagement: Implications for Adaptive Videogames and Online Training. *International Journal of Gaming and Computer-Mediated Simulations (IJGCMS)*, 7(1), 20-37.  
doi:10.4018/IJGCMS.2015010102
- Sherry, J. L. (2004). Flow and media enjoyment. *Communication Theory*, 14(4), 328–347.
- Thibault, R. T., Lifshitz, M., & Raz, A. (2016). The self-regulating brain and neurofeedback: experimental science and clinical promise. *Cortex*, 74, 247-261.
- Weibel, D., Wissmath, B., Habegger, S., Steiner, Y., & Groner, R. (2008). Playing online games against computer-vs. human-controlled opponents: Effects on presence, flow, and enjoyment. *Computers in Human Behavior*, 24(5), 2274-2291. doi: 10.1016/j.chb.2007.11.002
- Williams, D., Yee, N., & Caplan, S. E. (2008). Who plays, how much, and why? Debunking the stereotypical gamer profile. *Journal of Computer-Mediated Communication*, 13(4), 993-1018.
- Xue, S., Wu, M., Kolen, J., Aghdaie, N., & Zaman, K. A. (2017). Dynamic Difficulty Adjustment for Maximized Engagement in Digital Games. In *Proceedings of the 26th International Conference on World Wide Web Companion* (pp. 465-471). International World Wide Web Conferences Steering Committee.

# A Temporal Data-Driven Player Model for Dynamic Difficulty Adjustment

Alexander E. Zook and Mark O. Riedl

School of Interactive Computing, College of Computing  
Georgia Institute of Technology  
[{a.zook,riedl}@gatech.edu](mailto:{a.zook,riedl}@gatech.edu)

## Abstract

Many computer games of all genres pit the player against a succession of increasingly difficult challenges such as combat with computer-controlled enemies and puzzles. Part of the fun of computer games is to master the skills necessary to complete the game. Challenge tailoring is the problem of matching the difficulty of skill-based events over the course of a game to a specific player's abilities. We present a tensor factorization approach to predicting player performance in skill-based computer games. Our tensor factorization approach is data-driven and can predict changes in players' skill mastery over time, allowing more accurate tailoring of challenges. We demonstrate the efficacy and scalability of tensor factorization models through an empirical study of human players in a simple role-playing combat game. We further find a significant correlation between these performance ratings and player subjective experiences of difficulty and discuss ways our model can be used to optimize player enjoyment.

## Introduction

Many computer games of all genres pit the player against a succession of increasingly difficult challenges: combat with NPC enemies, puzzles, strategic planning and execution, etc. The player is expected to master a set of skills pertaining to the game mechanic over the course of the game. Some believe that this mastery of the game is a fundamental aspect to having fun in computer games (Koster 2005). However, contemporary computer games are being played by increasingly diverse audiences that differ in their skills and interests in games. This increasing variability in ability, speed of mastery, and growing diversity in tastes for game aesthetic and narrative content has prompted a recent growth of interest in automated methods to fit game content to these diverse abilities and interests. These efforts require both modeling the abilities and interests of players as well as adapting existing game content to those differences.

Many games revolve around *skill-based events*, periods of game play, such as combat or puzzles, in which the player must perform a specific skill. We see two main challenges to adapting computer games to fit individual player differences: *challenge tailoring* and *challenge contextualization*.

Challenge tailoring (CT) is the problem of matching the difficulty of skill-based events over the course of a game to a specific player's abilities. For example, in an action role-playing game such as *The Legend of Zelda* challenge tailoring may manifest as configuring the number, health, or damage dealt by various enemies at various times throughout the game. CT is similar to *Dynamic Difficulty Adjustment* (DDA), which only applies to online, real-time changes to game mechanics to balance difficulty. In contrast, CT generalizes DDA to both online and offline optimization of game content and is not limited to adapting game difficulty. Challenge contextualization (CC) is the related problem of constructing game events that set up the conditions for skill events and motivate their occurrence to the player. For example, the challenge of slaying a dragon may be contextualized by the dragon kidnapping a princess. Challenge contextualization includes common AI problems of quest generation, story generation in games, and interactive storytelling.

In this paper, we focus on the challenge tailoring problem in adventure role-playing games. Realizing challenge tailoring requires both a player model and an algorithm to adapt content based on that model. Effective player modeling for the purposes of challenge tailoring requires a data-driven approach that is able to predict player behavior in situations that may have never been observed. Because players are expected to master skills over time when playing a game, the player model must also account for temporal changes in player behavior, rather than assume the player remains fixed. Modeling the temporal dynamics of a player enables an adaptive game to more effectively forecast future player behavior, accommodate those changes, and better direct players toward content they are expected to enjoy. Further, forecasting enables player models to account for interrelations among sequences of experiences—accounting for how foreshadowing may set up a better future revelation or how encountering one set of challenges builds player abilities to overcome related challenges that build off of those.

In this paper we present and evaluate a temporal player modeling approach. We employ tensor factorization techniques to create temporal models of objective player game performance over time in a turn-based role-playing game and demonstrate the efficacy of our approach over related data-driven methods through comparisons from an empirical study. We model performance instead of difficulty be-

cause performance is objectively measurable while difficulty is subjective; we show difficulty and performance are significantly correlated for this particular domain. Finally, we suggest how our tensor factorization player model may be used for challenge contextualization.

## Related Work

Smith et al. (2011) overview the landscape of player modeling in computer games. In their taxonomy, we are investigating individual, empirical, generative player models. Research in player modeling has typically addressed the challenge tailoring problem either by developing purely behavioral models or relying on predictions that ignore temporal changes in player data. Hunicke and Chapman (2004) model players by computing the average and variance of player damage and item inventory. Dynamic Difficulty Adjustment is achieved via a hand-crafted policy prescribing actions to take based on player health and inventory states. Magerko et al. (2006) interactive story players using a vector of competence levels for various skills and associated confidence values. The system selects from a pool of challenges based on a best fit between the characteristics of the challenge event and the current state of the skill vector. van Lankveld et al. (2008) role-playing game players using their progress and health, dynamically adjusting sets of presented enemies to enforce a given level of player health over progress. In contrast, our data-driven modeling approach explicitly forecasts changes in player performance, combines information across players, and proactively constructs a long-term set of challenges based on these predictions.

Subjective self-report indications of challenge have also been used to dynamically tailor game play (Yannakakis and Togelius 2011). Pedersen et al. (2009) train a neural network to predict player self-reported experiences of fun, challenge, and frustration based on measures of player behavior and in-game content. Yannakakis, Lund, and Hallam (2006) employ a neural network to predict player self-reported interest. Our approach extends these models by correlating time-varying measures of performance to self-report measures, enabling forecasts of player experience forward in time.

While we believe our work is the first application of tensor factorization to challenge tailoring problems, we note that similar techniques have been used to model student performance over time on standardized tests (Thai-Nghe, Horvath, and Schmidt-Thieme 2011).

## Player Model

We explore tensor factorization techniques for modeling player performance in action role-playing games. While we focus on action role-playing games, we believe our techniques generalize to any games that make regular use of skill-based events. Tensor factorization techniques decompose multidimensional measurements into latent components that capture underlying features of the high-dimensional data. Tensors generalize matrices, moving from the two-dimensional structure of a matrix to a three or more dimensional structure. For our player modeling approach we extend two-dimensional matrices representing player perfor-

mance against particular enemy types to add a third dimension representing the time of that performance measure.

We chose to use tensor factorization due to its favorable scaling properties, ability to cope with missing data, high accuracy, speed in generating predictions, and previous success in other applications. Tensor factorization is an extension of matrix factorization, which has been used in collaborative filtering applications—such as the Netflix Prize data mining competition—to great success. Matrix factorization offers the key advantage of leveraging information from a group of users that has experienced a set of content to make predictions for what a new group of individuals that has only been partially exposed to that content will do. Specifically, user data is represented in a  $M = U \times I$  matrix indicating user preference ratings on items and decomposition extracts latent factors relating to users and items. Tensor factorization adds more dimensions, such as time, and extracts latent factors related to these other dimensions as well. Matrix and tensor factorization scale effectively to large numbers of users and items, handle missing information from users and achieve high accuracy (Koren and Bell 2011; Su and Khoshgoftaar 2009).

Formally, we represent player data in a tensor  $Z = U \times I \times T$ . In our simple spell-casting action role-playing game (described in the next section),  $U$  is the player (“user”),  $I$  is the spell type (“item”) and  $T$  is the time of the performance recording. CP decomposition is a generalization of singular value decomposition from matrices to tensors. In CP decomposition the three-dimensional  $Z$  tensor is decomposed into a weighted combination of three vector latent factors,

$$Z \approx \sum_{k=1}^K \lambda_k w_k \circ h_k \circ q_k$$

where  $\circ$  is the vector outer product,  $\lambda_k$  are positive weights on the factors,  $w_k$  are player factors,  $h_k$  are spell type factors, and  $q_k$  are time factors.  $K$  is the number of components used for each factor as an approximation of the true structure, keeping the set of the  $K$  most important components found (Kolda and Bader 2009). The decomposition can be computed by minimizing the root mean squared error between the factor inner product above and true data values, iteratively fitting each of the factors while fixing values of the other factors until convergence. We employ the N-way toolbox (Andersson and Bro 2000) to perform this decomposition. Predictions in this model consist of taking the inner product of these three factors, a computationally efficient process.

## Experiment

We focus on skill-based aspects of games that require procedural knowledge: the ability to correctly act in given circumstances, typically with limited time for decision-making. To test our tensor factorization model we conducted a study of player learning in a turn-based role-playing game. Below we present the game used for the study, study methodology, and the results of comparing our modeling technique to related approaches.



Figure 1: A battle between monsters and the player team.

Table 1: Spell Effectiveness Matrix.

| Attack ↓ Def. → | fire | water | acid | ice | light. | earth | force | undeath |
|-----------------|------|-------|------|-----|--------|-------|-------|---------|
| fire            | 1    | 0     | 1    | 2   | 1      | 2     | 1     | 0       |
| water           | 2    | 1     | 0    | 1   | 0      | 1     | 2     | 1       |
| acid            | 1    | 2     | 1    | 0   | 1      | 0     | 1     | 2       |
| ice             | 0    | 1     | 2    | 1   | 2      | 1     | 0     | 1       |
| lightning       | 1    | 2     | 1    | 0   | 1      | 0     | 1     | 2       |
| earth           | 0    | 1     | 2    | 1   | 2      | 1     | 0     | 1       |
| force           | 1    | 0     | 1    | 2   | 1      | 2     | 1     | 0       |
| undeath         | 2    | 1     | 0    | 1   | 0      | 1     | 2     | 1       |

## Game Domain

We implemented a turn-based role-playing game in which the player leads a group of four characters through a sequence of spell-casting battles against groups of enemy monsters. For the purpose of this study, we ignore the quest-like contextualization of the battles. See Figure 1 for the game’s battle interface. Turns are limited to 10 seconds to require mastery of a complex spell system.

Each enemy is associated with one of eight spell types and player-controlled characters can attack with four of the eight possible spells. Casting a particular spell against an enemy of a particular type results in an attack being effective, ineffective, or super-effective, resulting in normal damage, no damage, or double damage against an enemy (see Table 1).

We intentionally created a spell system that was difficult to completely memorize, but contained intuitive combinations—water spells are super-effective against fire enemies—and unintuitive combinations—undead spells are super-effective against force enemies—ensuring that skill mastery could only be achieved by playing the game. Note that pairs of spells—e.g., fire and force—are repeated in Table 1. This ensures a simpler underlying skill structure for players to learn; there are effectively only four spells. A scoring system based on spell effectiveness motivates players to learn; effective spells earn two points, ineffective spells earn zero points, and super-effective spells earn five points. Enemy attacks decrease player score by one. Player characters were assigned different spell sets, forcing players to learn the spell system.

## Study Methodology

We recruited 32 participants to play our role-playing game, which was simplified to present a series of battles one after another, as in Figure 1. In our study we first gave players five minutes to review a text document explaining the spell system and the game interface. Once familiar with the game, players completed the sequence of eleven battles while we recorded the performance of the player in terms of effectiveness of spells chosen against enemies. After each battle we additionally asked players to report how difficult and enjoyable the battle was on a 5-point Likert scale.

We recorded each spell players cast on each enemy on each turn and battle in the game along with the associated performance value: 0 for ineffective, 1 for effective, and 2 for super-effective. Because spell effectiveness determines damage to enemies, player behavior traces varied in the number of turns taken, but had the same number of battles. We average performance for each spell type across all turns within a given battle, leaving the performance value missing for any spells not used in the battle.

We hypothesize that a tensor factorization approach will outperform non-tensor approaches. The tensor factorization model organizes player data in a three-dimensional tensor (player, enemy spell type, battle number), where each point in the tensor is the average performance of the player in attacking foes of a given spell type during a given battle. Spell types not used in a given battle were recorded as missing values. This produces a  $32 \times 8 \times 11$  tensor for our 32 players, 8 spell types, and 11 battles. We compared the tensor model to two other models: matrix factorization using an unfolded tensor, and matrix factorization averaging over time. The matrix unfolding of this tensor simply concatenates battles column-wise, producing a  $32 \times 88$  matrix recording player performance on each spell type and battle number combination. The final matrix factorization approach averaged player performance against spell types across all battles, removing any temporal information. Data points represent average player performance over their entire battle history against an enemy type. If our hypothesis is confirmed, then the tensor model captures additional structure lost in the unfolded matrix model when all time points are concatenated together.

We also hypothesize that there is an inverse relationship between objective, measurable player performance and subjective, self-reported difficulty. Should this hypothesis hold it will verify that in the context of action role-playing games we can use skill performance as a proxy for difficulty.

## Results

We first compared players according to a variety of collected demographic information (age, gender, race, ethnicity, prior video game and role-playing game experience) and found no significant differences among these groups in our data set, validating the use of a single model across all players. We measured the 10-fold cross-validated *percent variance explained* of the tensor, matrix unfolded, and time-averaged models. Percent variance explained is an error measure that compares the total variation in the data across cases with the variation remaining after a model has been applied to the

data. It describes how well a model captures variations in the data, with higher percentages indicating more powerful explanations.

The tensor model outperforms the other techniques for three or more components (see Table 2). While the tensor model does not achieve substantially better performance than the unfolded or time-averaged models on few components it shows much greater performance on larger numbers of components, reflecting its greater capacity to model complex underlying structure in the data. Notably, the tensor model shows comparable high performance for three, four, or five factors, indicating the spell matrix reflects an underlying structure testing the corresponding number of skills. As noted earlier our spell matrix actually only contains four spells (each spell has two names). This intuitively explains a result suggesting four underlying factors—the four unique spells—with similar numbers of factors reflecting a moderate amount of noise around this underlying information. The 100% score for the time-averaged model with 8 factors reflects the fact that this model is modeling 8 spell types with 8 factors and thus can trivially recover the same model.

| components | tensor | unfolded | timeavg |
|------------|--------|----------|---------|
| 2          | 92.59  | 90.82    | 95.86   |
| 3          | 92.18  | 89.44    | 72.41   |
| 4          | 88.72  | 86.30    | 39.99   |
| 5          | 90.11  | 61.77    | 45.02   |
| 6          | 89.11  | 63.66    | 24.28   |
| 7          | 87.11  | 24.29    | 36.30   |
| 8          | 80.05  | -10.81   | 100.00  |

Table 2: Comparison of percent variance explained by different models using varying numbers of factors.

We evaluated the scaling of the tensor model in three ways: (1) how well the model scales with additional data; (2) how well the model forecasts into the future for players; and (3) how well the model scales in the number of players used by the system. In all three cases we use 10-fold cross-validated percent variance explained averaged over 10 repetitions of the experiment. Our first assessment hid a random subset of all our data and generated predictions for those missing values as shown in Figure 2. Model performance increased with the amount of total data used, with the simple two-component model showing high performance across all amounts of data while the more complex six- and eight-component models required approximately 60% of the data to achieve high performance. Thus, tensor factorization techniques perform well with this kind and amount of data, with more complex models requiring additional data but achieving high performance with more information.

Our second assessment evaluated the accuracy of the tensor model predictions on future events by hiding all future battles for a group of target players (Figure 3). The simple two-component tensor model achieved high performance regardless of the number of battles used, while the more complex six- and eight-component models required approximately seven battles to achieve comparable performance.

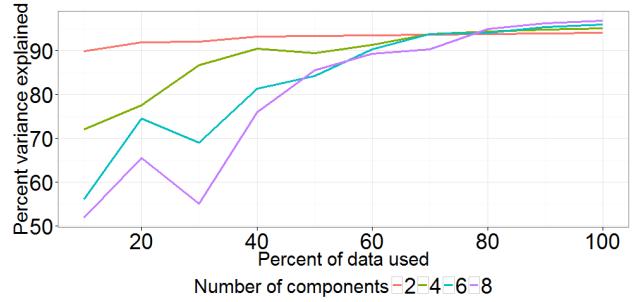


Figure 2: Percent variance explained of the tensor model as a function of the amount of data randomly subsampled from the data set, averaged over 10 repetitions.

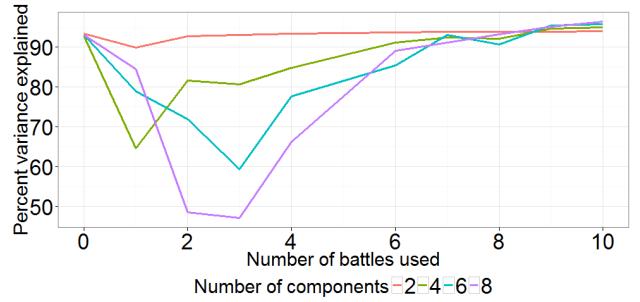


Figure 3: Percent variance explained of the tensor model when using first subset of battle data from randomly selected players, averaged over 10 repetitions.

Thus, tensor factorization models can generate useful forecasts after only a small number of battles observed for any given player. From a game design perspective, this suggests a relatively short training level in which the player can demonstrate skills can yield reasonable predictive power.

Our third assessment examined the tensor model accuracy when training and testing on varying numbers of players. We randomly subsampled from our full data set to use 6, 11, 16, 21, or 27 players and performed the same cross-validation analysis as above. Accuracy improved on the 2 component model from 81.17% variance explained with 6 players to 86.52% with 27 players, while the 4 component model improved from 52.55% to 86.29%, respectively. Other numbers of components showed similar trends, converging to approximately 86% accuracy with 27 players. The results of these three assessments demonstrate the power of the tensor factorization technique to scale with additional players and additional data for those players.

Finally, we found a significant correlation between objectively measure performance and subjectively experienced difficulty. Performance levels significantly differed (Kruskal-Wallis test for analysis of variance,  $p < 0.001$ ) between subjective difficulty ratings. A Kruskal-Wallis test assesses whether responses (here performance) in different groups (difficulty ratings) shows significant differences—it is a non-parametric alternative to the standard analysis of variance tests that assume the data has a Gaussian distri-

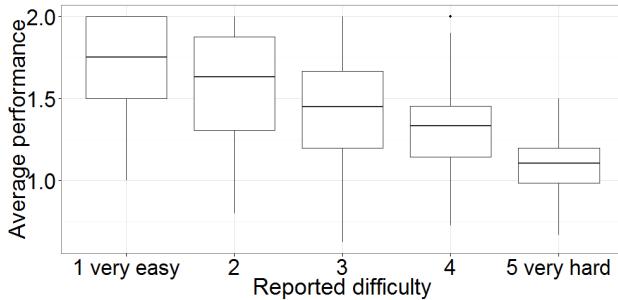


Figure 4: Comparison between objective performance values for subjective difficulty ratings, showing median values and the range from 25<sup>th</sup> and 75<sup>th</sup> percentiles.

bution, as our data showed skewing in performance measures for different response groups. A Dunnett’s test found all adjacent pairs of difficulty ratings significantly differed ( $p < 0.05$ ) in performance value, ranging from  $-0.11$  to  $-0.22$  (Figure 4). Dunnett’s test applies a more powerful version of the Student’s t-test to performance multiple comparisons among groups. Thus our second hypothesis is confirmed; objective measures of skill performance are inversely related to perceived difficulty.

## Content Adaptation

Our temporal player model can predict player performance on skill-based events in the future. To perform challenge tailoring of a game, the player model must be combined with information about how to use the model. In the next sections we describe (1) a target expected performance to compare predictions of an individual user’s performance against, and (2) an algorithm to select and/or parameterize skill-based events to bring predicted player performance in line with target performance.

## Target Performance Curve

We expand upon the concept of a *performance curve* (Zook et al. 2012), as an indication of the desired player performance over a sequence of skill-based challenges. In our game this indicates desired player performance across a sequence of battles, provided by the human designer. Figure 5 shows an example of a performance curve. This particular curve seeks a decrease in performance over time until the very end of the game. This game should appear to a player to increase in difficulty from challenge to challenge, even as the player continues to master the skills involved. Other curves are possible as well. A curve expressed by  $p = c$  (a horizontal line at a fixed constant,  $c$ ) indicates a game in which the difficulty appears to remain the same, even as the player’s skills improve. That is, the challenges may be parameterized to be more difficult, but because the challenge level is increasing at the same rate as player skill mastery, there should be little or no perceived change in difficulty. More complicated patterns, such as a series of rises and falls, can express complex designer intentions.

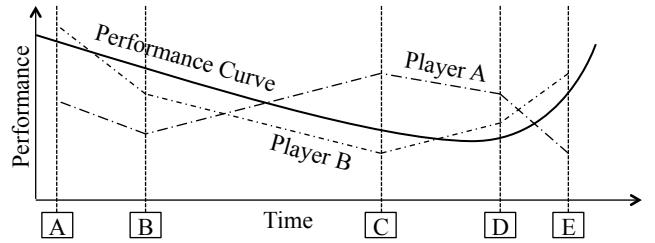


Figure 5: An example performance curve indicating a game that gets progressively more difficult until the very end, even as the player’s skills improve. Dotted lines indicated predicted performance for two players.

Note that performance curves are specifications of desired player performance, rather than difficulty. Although our studies show that performance and difficulty are inversely correlated, there may be other aspects of the game—such as ambiguity in game information—that are not necessarily captured in the performance metric. A performance curve bypasses the ambiguity of subjective difficulty while enabling designers to specify how they desire a particular performance metric to vary over the course of an experience.

## Challenge Tailoring and Contextualization

Given a performance curve and a temporal player model, challenge tailoring is the problem of selecting and spacing a sequence of skill-based events—in our case, battles—that match the predicted performance over time to the desired performance. This is a discrete optimization problem—battles are discrete units and the goal is to minimize the difference between predicted and desired performance values for these battles. A variety of techniques may be applied to solve these problems including constraint satisfaction, dynamic programming, and heuristic search techniques such as genetic algorithms (Smith and Mateas 2011; Togelius et al. 2011; Sorenson, Pasquier, and DiPaola 2011; Zook et al. 2012). In contrast to the reactive, near-term changes typically employed in DDA (Magerko, Stensrud, and Holt 2006; Hunnicke and Chapman 2004), temporal player models are able to also proactively restructure long-term content to optimize a global player experience.

Challenge contextualization is the problem of selecting non-skill-based events that explain, motivate, or justify the skill-based events. Quest generation and narrative generation techniques may be used to provide this contextualization of skill-based events, although other, non-narrative contextualization may exist as well. Challenge tailoring and challenge contextualization may be performed in a pipelined fashion or in parallel. Pipeline techniques afford modular and compositional approaches to tailoring game content. In particular, if tailoring of skill-based events takes precedence over contextualization such that contextualization (i.e., the narrative, quest, or mission) can be sub-optimal then one might choose to solve the discrete optimization problem of selecting and spacing all skill-based events before “filling the gaps” with non-skill-based, contextualizing events (cf., (Magerko, Stensrud, and Holt 2006)). Fixing the skill-based

events prior to contextualization makes challenge tailoring easier, but constrains the searchable context space.

Parallel techniques, conversely, may explore the full space of joint skill- and non-skill combinations, but trade this flexibility for greater complexity in the tailoring task. For example, we describe a technique that searches for a complete sequence of skill- and non-skill-based events that simultaneously optimizes for player performance and context (Zook et al. 2012). The question of whether to use a pipeline versus parallel approach depends on the importance of contextualization relative to skill tailoring and the extent to which skill- and non-skill-based events appear seamless.

## Conclusions

As computer games grow in popularity personalization of game content—which we cast as the paired problems of challenge tailoring and challenge contextualization—will also grow in importance. Unlike many other player modeling approaches to challenge tailoring and dynamic difficulty adjustment, we use a data driven approach that explicitly incorporates a temporal component. This temporal component allows us to more accurately forecast *future* player performance by modeling changes in a player’s skills.

A performance curve provides authorial control over the game in the form of a target level of performance. Since objective performance is inversely correlated with subjective difficulty, a performance curve can guide an algorithm in the selection and parameterization of skill-based events over time. The tensor factorization model gives a system insight into how the player will perform on various sequences of challenges. With these tools, there are many discrete-optimization algorithms that can solve the challenge tailoring problem. Challenge contextualization, though outside the scope of this paper, can further ensure challenges make sense to the player while promoting player skill mastery.

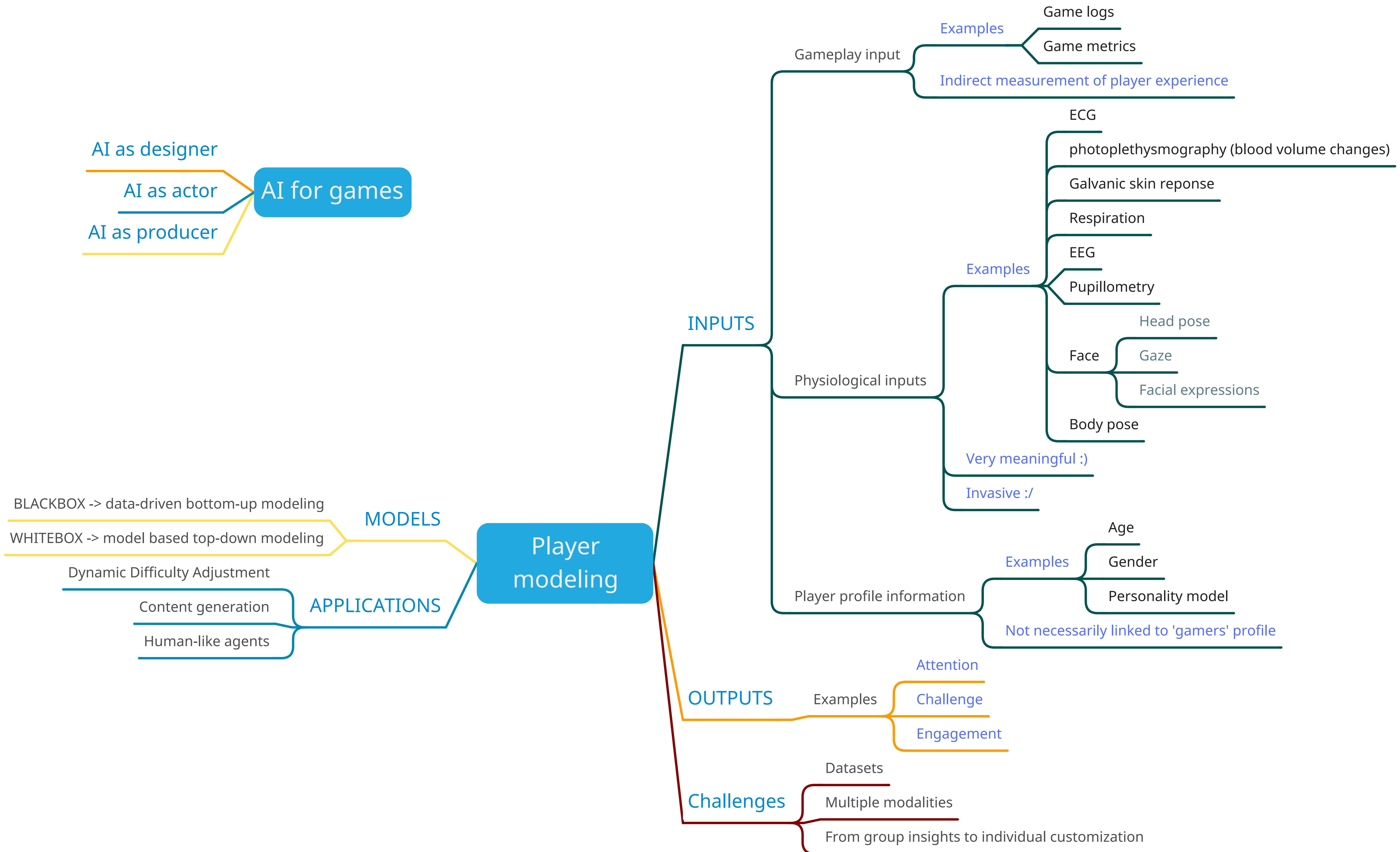
Mastery of skills is correlated with fun (Koster 2005). A game that is able to tailor its difficulty to meet the abilities of the player may be perceived as more enjoyable to a wider range of players because it is never unintentionally boringly easy or frustratingly hard. This in turn has the potential to increase game replayability and make certain games accessible to a wider diversity of players.

## Acknowledgments

The project or effort described here has been sponsored by the U.S. Army Research, Development, and Engineering Command (RDECOM). Statements and opinions expressed do not necessarily reflect the position or the policy of the United States Government, and no official endorsement should be inferred.

## References

- Andersson, C., and Bro, R. 2000. The N-way toolbox for MATLAB. *Chemometrics and Intelligent Laboratory Systems* 52(1):1–4.
- Hunicke, R., and Chapman, V. 2004. AI for dynamic difficulty adjustment in games. In *Proceedings of the AAAI Workshop on Challenges in Game Artificial Intelligence*.
- Kolda, T., and Bader, B. 2009. Tensor decompositions and applications. *SIAM review* 51(3):455–500.
- Koren, Y., and Bell, R. 2011. Advances in Collaborative Filtering. In Ricci, F.; Rokach, L.; Shapira, B.; and Kantor, P. B., eds., *Recommender Systems Handbook*. Boston, MA: Springer. 145–186.
- Koster, R. 2005. *A Theory of Fun in Game Design*. Paraglyph press.
- Magerko, B.; Stensrud, B.; and Holt, L. 2006. Bringing the schoolhouse inside the box - a tool for engaging, individualized training. In *Proceedings of the 25th Army Science Conference*.
- Pedersen, C.; Togelius, J.; and Yannakakis, G. N. 2009. Modeling Player Experience in Super Mario Bros. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games*.
- Smith, A., and Mateas, M. 2011. Answer set programming for procedural content generation: A design space approach. *IEEE Transactions on Computational Intelligence and AI in Games* 3(3):187–200.
- Smith, A.; Lewis, C.; Hullett, K.; Smith, G.; and Sullivan, A. 2011. An inclusive view of player modeling. In *Proceedings of the 6th International Conference on Foundations of Digital Games*.
- Sorenson, N.; Pasquier, P.; and DiPaola, S. 2011. A Generic Approach to Challenge Modeling for the Procedural Creation of Video Game Levels. *IEEE Transactions on Computational Intelligence and AI in Games* 3(3):229–244.
- Su, X., and Khoshgoftaar, T. M. 2009. A Survey of Collaborative Filtering Techniques. *Advances in Artificial Intelligence* 2009:1–19.
- Thai-Nghe, N.; Horvath, T.; and Schmidt-Thieme, L. 2011. Factorization Models for Forecasting Student Performance. In *Proceedings of the 4th International Conference on Educational Data Mining*.
- Togelius, J.; Yannakakis, G.; Stanley, K.; and Browne, C. 2011. Search-Based Procedural Content Generation: A Taxonomy and Survey. *IEEE Transactions on Computational Intelligence and AI in Games* 3(3):172–186.
- van Lankveld, G.; Spronck, P.; and Rauterberg, M. 2008. Difficulty scaling through incongruity. In *Proceedings of the 4th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*.
- Yannakakis, G. N., and Togelius, J. 2011. Experience-Driven Procedural Content Generation. *IEEE Transactions on Affective Computing* 2:147–161.
- Yannakakis, G.; Lund, H.; and Hallam, J. 2006. Modeling children’s entertainment in the playware playground. In *Proceedings of the 2nd IEEE Symposium on Computational Intelligence and Games*.
- Zook, A.; Riedl, M. O.; Holden, H. K.; Sotilare, R. A.; and Brawner, K. W. 2012. Automated scenario generation: Toward tailored and optimized military training in virtual environments. In *Proceedings of the 7th International Conference on the Foundations of Digital Games*.



# AI for Game Production

Mark Owen Riedl

School of Interactive Computing  
Georgia Institute of Technology  
riedl@cc.gatech.edu

Alexander Zook

School of Interactive Computing  
Georgia Institute of Technology  
a.zook@gatech.edu

**Abstract**—A number of changes are occurring in the field of computer game development: persistent online games, digital distribution platforms and portals, social and mobile games, and the emergence of new business models have pushed game development to put heavier emphasis on the live operation of games. Artificial intelligence has long been an important part of game development practices. The forces of change in the industry present an opportunity for Game AI to have new and profound impact on game production practices. Specifically, Game AI agents should act as “producers” responsible for managing a long-running set of live games, their player communities, and real-world context. We characterize a confluence of four major forces at play in the games industry today, together producing a wealth of data that opens unique research opportunities and challenges for Game AI in game production. We enumerate 12 new research areas spawned by these forces and steps toward how they can be addressed by data-driven Game AI Producers.

## I. INTRODUCTION

Over the past several years the game industry has undergone a series of major changes. The increasing prominence of persistent online games, digital distribution platforms and portals, mobile and social games, and the emergence of new business models have all changed fundamental aspects of making and playing games. Developers increasingly focus on the live operation of a game, rather than creating a boxed and finalized product. Players are more diverse, have access to games in more places and at more times, and produce more data and content for developers to leverage than ever before. Across these changes four forces have come to the fore:

- 1) Games and cross-game play is increasingly persistent and presenting longer-term experiences
- 2) Game developers are starting to see their game titles as an ecosystem wherein players may move from game to game
- 3) Player communities and social gameplay have gained prominence
- 4) Developers and players have a growing interest in coupling the real world and virtual world(s)

Some of these forces have been around for a number of years, while others are just beginning to emerge. The consequence of these forces is a profound opportunity for Artificial Intelligence (AI), Computational Intelligence (CI), and Machine Learning (ML) in games (collectively referred to as *Game AI*) to play an even greater role in the development of computer games and the delivery of engaging real-time experiences to players.

Through these forces we see an opportunity for Game AI to address new research questions that have the potential to dramatically impact game development practices. The most

significant consequence of the shifting landscape of computer game development is the generation of massive amounts of data. Researchers and game developers can leverage this data in a new paradigm of data-driven Game AI focused on how to use these sources of data to improve game development practices and to deliver more engaging experiences. However, we are not merely advocating that researchers and game developers adopt data-driven analogues to existing Game AI practices (although that would be a worthy endeavor). Instead, we are proposing that the market forces enumerated above are forcing the industry to change how they think about game development processes. The modern development environment presents significant challenges to scalability—of both the practice of creating games and also the size and scope of games themselves—that are readily addressed by artificial intelligent, computational intelligence, and machine learning research.

Game AI was initially about supporting the interaction between player and the game itself. Recently there has been a push for procedural content generation as a means to support and augment the game developer on a game by game basis. In this paper, we present our desiderata on the future of Game AI in which intelligent systems support the entire game production pipeline from game creation to live operation, and across a number of games and game genres. This perspective of Game AI for production does not supplant or replace prior perspectives on Game AI, but presents a new lens through which to see an expanded role for Game AI in computer game production. We will enumerate 12 novel research questions that arise when considering the role of AI as Producer and discuss steps toward how these challenges may be addressed.

## II. THE SHIFTING LANDSCAPE OF GAME DEVELOPMENT

In this section, we enumerate some of the prominent new forces that are shaping the way that computer game development companies create games. Many of these forces have arisen as a confluence of advances in computing technology and market forces that result from a maturation of the computer game industry.

*1) Persistent Games:* The rise of Massively Multiplayer Online Games (MMOGs), social games, online game portals and long-term or recurring game experiences is creating long-term data on players within games. Many games have players join, play over long periods of time, and potentially rejoin at later times. Developers are increasingly pressed to develop content that provides long-term engagement with a game, rather than a closed experience with clear beginning and end.

*2) Ecosystem of Games:* Developers have a wider variety of tools to build games and lowered barriers to distributing

games to new users through digital platforms and online portals. Together this puts greater emphasis on keeping players engaged within an ecosystem of games from a single developer, rather than focusing on experiences only within one game. Developers are driven to provide new content of multiple kinds (including meta-game objectives) that engage players across games. The growing diversity of game players has led to additional emphasis on ensuring an ecosystem of games provides a diverse range of experiences.

*3) Player Communities:* Player communities have emerged as a major driving force for the success (and failure) of games. Players generate a mass of content from reviews and walkthroughs through add-ons and full game modifications. Continually engaging communities, supporting player socializing within the community, managing how players impact one another's experiences (positively and negatively), and leveraging user-generated content are growing concerns.

*4) Coupling of Real and Virtual Worlds:* Games are more widely adopting ways to connect to the real world. Sensing systems from the Kinect and Wiimote to mobile phone GPS provide more data on players' real-world environment. Output modalities from second screen experiences and mobile phone augmented reality to virtual reality are emerging as new ways to view game content. At the same time these technologies have introduced a host of challenges around how games can interface with and use this real-world context and respond to more unstructured types of information.

Each of these forces presents new opportunities to solve problems of real-world applicability to game developers. A side effect of each of these forces is the massive amounts of data being generated about game players. While it is not necessarily the case that new research problems will require data-driven AI, CI, and ML techniques, we see this data as a tool for tackling real-world game development problems.

### III. BACKGROUND: THREE ROLES OF GAME AI

In 2001, Laird and van Lent [1] put forth their seminal argument for AI in computer games as an academic pursuit. They specifically argued that the pursuit of “human-level” AI systems could use computer games as testbeds for research because games were intermediate environments between the toy domains researchers had been using and the full complexity of the real world. While this opened Game AI as an academic endeavor, the automation of various aspects of computer games has been a part of the industrial practice of creating commercial computer games since the beginning.

*Game AI* has come to refer to the set of tools—algorithms and representations—developed specifically to aid the creation and management of interactive, real-time, digital entertainment experiences. While games are played by humans, there are a number of aspects of the game playing experience that must be automated: roles that would be best performed by humans but are not practical to do so:

- Opponents and enemies that are meant to survive for only a short time before losing.
- Non-player characters in roles that are not “fun” to play such as shopkeepers, farmers, or victims.

- Companions in single-player experiences and non-player characters in support roles.
- Drama management at scale.
- Game designer for personalized experiences at scale.

As we go down this list, Game AI is charged with taking progressively more responsibility for the quality of the human player's experience in the game.

We group Game AI approaches into three broad roles that AI, CI, or ML takes in the creation and delivery of engaging entertainment experiences. Each role targets a different stakeholder: players, designers, and producers. The first role is *AI as Actor*, in which the Game AI system mediates between the player and the game to create a compelling real time experience. The most common manifestation of AI under this paradigm is controlling or managing bots and non-player characters. The second role is *AI as Designer*, in which the AI mediates between a game designer and a single player-game system. Under this paradigm, AI systems might procedurally generate game content or adapt the game to particular players to scale the game development process. Finally, we propose a third role, *AI as Producer*, in which the AI mediates between game producers and a number of systems of players, designers, and games. Figure 1 shows how the different roles relate to each other and the three primary human stakeholders.

These three roles are not distinct phases in the pursuit of Game AI, but overlapping sets of concerns and driving problems, all of which need to be pursued individually or in unison. We see AI Producers as a superset of AI Designers, encompassing a broader set of research questions. Equivalently, we see this as a shift from Game AI for game design to Game AI for game production. In industry the differences between designers and producers have blurred as technical barriers to content production have lowered and gameplay and business (e.g. monetization and marketing) are more tightly coupled.

#### A. Artificial Intelligence as Actor

Historically, the earliest uses of artificial intelligence in computer games was to mediate between users and the game. AI served the role of an artificial human opponent or playmate, enabling play without requiring other people or filling roles humans would be loathe to fill in a game. Compared to non-Game AI, the intelligence built into games places a greater emphasis on creating engaging and entertaining experiences for users, rather than maximizing a utility function such as score or win/loss rates.

For the AI as Actor role, research has focused on non-player character (NPC) path planning and decision making. Game agent decision making emphasizes the believability of characters to support the suspension of disbelief that the player is interacting with software instead of a monster, human opponent, or human companion. These problems are still being pursued by industry developers and academic researchers.

*Drama management* is another way for AI to mediate between users and the NPCs and other aspects of a game or virtual world [2], [3], [4]. A drama manager is an omniscient agent responsible for delivering an enjoyable and coherent narrative experience to players, much as a “Dungeon Master” does the same for tabletop role playing games.

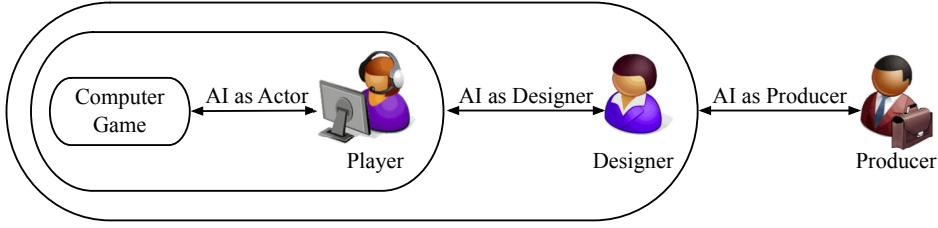


Fig. 1. Three roles of Game AI and their human stakeholders. *AI as Actor* describes how AI mediates between the player and the game itself. *AI as Designer* describes how AI mediates between a single game designer and the human-computer system of player and game. *AI as Producer* is a new role describing how AI mediates between producers and many systems of designers, users, and games.

### B. Artificial Intelligence as Designer

The second role of Game AI is to mediate between the human designer (or developer) and the human-computer system comprised of game and player. In our metaphor, game designers are responsible for building and defining a game, analyzing how players interact with the game, and iteratively refining a game to achieve a design vision. This paradigm for artificial intelligence is often referred to as *Procedural Content Generation* (PCG)—algorithms and representations for generating any and all components of games [5], [6], [7]. Offline content generation emphasizes producing content designers may curate or refine as a means of increasing the efficiency of the game development process. Online, or just-in-time, generation focuses on providing larger amounts of variation than human designers can achieve alone and/or tailoring content to a given user. A primary concern for PCG researchers has been (a) ways to appropriately represent game content to suit generation algorithms, while (b) providing means for users to interact with generation systems to author desired content and outcomes.

*Game adaptation* combines content generation and player modeling to enable AI designers to tailor games to individual players, emphasizing a closed loop of modeling player actions and automated adjustment of game content based on design goals for player behavior [8], player skill acquisition [9], or maximizing player enjoyment [10], [11], [12]. Game adaptation taken to its extreme introduces the problem of *game generation*—the automatic creation of entire games driven by (real or simulated) player feedback.

The availability of large data sets is having an impact on those who see AI as designer. In particular, *game analytics* involves capturing, aggregating, understanding, and visualizing player behavior to support designer understanding [13]. Player modeling research examines methods to describe and predict player behavior, potentially to be used by designers or automated AI systems in response [14]. While not a traditional domain of Game AI, player modeling has become increasingly prominent as AI agents shift to learning to adapt to players. Machine learning (e.g. [15], [16]) and evolutionary computing [17] are the primary areas currently being employed to perform these modeling tasks.

### C. Artificial Intelligence as Producer

The third role of Game AI uses a metaphor of AI as game producer. In our metaphor, producers concern themselves with the entire set of games and game content being made by a company, along with related aspects of managing

player communities. AI Producers mandate a shift from single player experiences within a closed game to long-term player experiences within an open game, understanding a player across multiple games in an ecosystem, and understanding how multiple players interact as an in-game and out-of-game community. AI Producers extend many methods of AI designers, driving a shift to model and adapt games that distinguishes characters (in-game avatars or personas) from players (agents manipulating those characters).

AI Producers also leverage a broader sense of context for enhancing game experiences including the community of players and the real-world context of their activities. AI for game production puts a premium on broadening the scope of Game AI to better integrate the increasingly pervasive nature of games into how games entertain and engage users. Games no longer are sharply bound by a single delivered product and Game AI ought to respond by incorporating these newly opened borders for new research domains. Effectively responding to these challenges will require new methods to leverage the masses of data being produced by players to improve interactive experiences.

## IV. GAME AI PRODUCER: RESEARCH QUESTIONS

AI producers will use the wealth of data being generated to meet the needs imposed by the four forces of (1) persistent games, (2) game ecosystems, (3) player communities, and (4) real-virtual world coupling. In each case these forces provide unique opportunities for data-driven approaches to Game AI that enhance player experience through richer sources of information and emerging modes of game-related interaction. Rather than replace earlier roles and stakeholders of Game AI, we assert that the new role of AI as Producer must emerge to address the novel concerns raised by the four forces. This new role for Game AI enhances and extends the AI techniques defined from earlier roles, leading to new research questions and techniques that potentially overlap multiple roles. In the next sections, we revisit the forces and the research questions that accompany them. Due to the overlapping nature of roles, in many cases research questions may address more than one role simultaneously.

### A. Persistent Games

Persistent games shift from closed games comprising time investments typically on the order of days to ongoing and extended game experiences spanning months or years generating long-term data on player history and activities. AI Producers address the full lifetime of a player, spanning the standard

business concerns of acquiring new players, retaining players over a long period of time, and reacquiring lapsed players. Key research questions around using long-term data to improve player engagement focus on: lifelong agents, gameplay support, and engagement-oriented content generation. Unifying these is a view of Game AI providing for long-term player engagement across an increasingly diverse group of players.

*1) How can an AI agent interact with players over very long periods of time?* An agent that persists for months, years, or even a user's lifetime is referred to as a *lifelong agent*. In the context of computer games, lifelong agents are NPCs that learn about you as a player over time. Lifelong agents serve as long term companions (or adversaries) that recognize and adapt to changes in players over time and use historical interactions with players to shift their behavior. That is, lifelong agents get to know players, and become familiar with the player, and familiar to the player.

Lifelong agents are relatively unexplored in the domain of games. In the domain of virtual agents for healthcare, Bickmore and colleagues have been designing and developing agents that interface with humans longitudinally [18]. In the context of games, lifelong agents may foster player empathy for companions—and enmity for rivals—or engage users in social interactions with game world NPCs. In addition, adapting agent behaviors to foster long-term engagement stands as a key to the problem of many online games face in creating vibrant and stable communities of players.

A central challenge for lifelong agents will be adapting to games with continually evolving content in the form of patches, expansions, downloadable content, and other incremental updates to the game the agent inhabits. Research in *lifelong machine learning* investigates how agents can transfer knowledge between particular tasks, continually learn and refine knowledge, uncover representations for complex information, and incorporate guidance or feedback from humans [19]. Addressing these challenges for game agents can provide enormous benefits to developers of live and ongoing games.

*2) How can a Game AI system support deep gameplay?* A key to persistent games is player retention. Many contemporary games have high ceilings for player skills to build up to—through complex underlying systems or ever-evolving multiplayer competition. However, complex games often lose players early in the game, especially as players increasingly have more diverse backgrounds and skills. Gameplay support agents act as mentors to players to help them overcome challenges that might otherwise cause them to quit playing a game. Gameplay support AI observes players, learns their gameplay strengths and weaknesses, and intervenes to provide players with appropriate hints, training materials, or content adjustments as needed. For example, if a player shows an inability to counter a particular strategy in an RTS the AI would identify the missing player skills and could provide instruction about the appropriate response, training video demonstrations, or set up game scenarios to practice the requisite skill. At the extreme a gameplay support agent could coach players to climb their way to the top of multiplayer competition.

Intelligent Tutoring Systems [20] present one vision for gameplay support. Applying interactive tutors to support long-term player engagement will need advances in representing

player skills, modeling players based on their game activities, devising interventions to improve those skills, and adjusting those interventions in response to tutoring success or failure. Ultimately, gameplay support systems should aspire to automatically generate tutorials—for introducing new content through coaching players along difficult missions—based on game features and gameplay data. Extracting examples appropriate to demonstrate skills from gameplay data, recognizing player skills, and choosing appropriate timing and presentation style for tutorial information are all key challenges to apply gameplay data toward automated game tutoring.

Another technique for gameplay support might lie in *Dynamic Difficulty Adjustment* (DDA). Dynamic difficulty adjustment systems make real-time adjustments to game parameters, item placement, enemy behavior, and other content to suit player abilities. Techniques for DDA have involved classical cybernetic systems [21], production systems [22], optimization of generator output from neuro-evolutionary [12] or machine learning [23] systems, or logic programming on possible player behavior [8]. Adapting these techniques for gameplay support requires capturing the long-term effects of interventions on player engagement [24], richer models of player skills related to various gameplay domains, and techniques to intelligently reuse high quality human-authored content where possible.

*3) How can a Game AI system generate motivational game content?* Content generation for long-term engagement models player values, preferences, and motivations to generate content that continues player engagement over long periods of time. The goal is to improve player retention or reacquire players who have lost interest in a game. Whereas gameplay support addresses potential “pain points” of gameplay to prevent player early dropout and improve player acquisition, long-term motivational content generation focuses on how to best retain players once they have committed to a game and encourage players who have lapsed from a game to return.

Compared to previous work on content generation, motivational content generation should incentivize players to take advantage of aspects of a game they already enjoy or to explore new elements of a game they might not have encountered. For example, generation of personalized achievements can encourage players to try alternative ways to complete a mission or level. Likewise, a system might generate mini-games within the context of a larger, persistent games based on the behaviors that a player already favors. Other forms of motivational content may exist. Regardless, a system must use longitudinal player data to determine what content to create, when to create it, and what value the content will provide to different players.

Drama managers are disembodied virtual agents that monitor virtual worlds and intervene to drive a narrative forward based on models of player experience quality [4]. Drama managers implemented in *Left 4 Dead* and *Darkspore* have demonstrated the ability to increase game replay value by varying game content and modulating player intensity levels. Drama managers that procedurally generate narratives have been demonstrated in the context of short-term experiences. Extending these systems to motivate persistent game players through narratives requires further work to personalize narratives to players through data [10], create a potentially infinite variety of narratives or quests [25], and chain narratives to

create a persistent, coherent sense of progression in an ever-growing game story.

## B. Ecosystem of Games

Digital distribution and online game portals have led to increasingly diverse games and a growing long-tail of smaller games appealing to niche interests. Ecosystems of games push entertainment goals to players' experiences across a number of potentially unrelated games, rather than within a single game. Connecting characters and gameplay from a single player across multiple games opens new opportunities in cross-game agents, cross-game content generation, and automated online game designs experiments. Previous player modeling and content generation research can play new roles when players interact with many distinct games, requiring advances in representing, collecting, and reasoning on game design knowledge.

### 4) How can AI agents interact with players across games?

Starting from the premise that lifelong virtual agents learn how to interact with individual players, we speculate that it may be advantageous for characters to interact with their players *across* many games in a company's ecosystem. To some degree, the solution involves designing for cross-game characters that manifest as recurring characters or playmates who join players across many games. However, as a lifelong agent adapts to an individual player, the question becomes one of how an agent with a constantly adapting personality, memory, and history of interactions with a player can manifest its individualized nature in the context of new games.

Cross-game agents can draw from research on competitive cross-game AI and work on socially present agents. The general game playing competition has spawned research on representing and reasoning on generic game state and rule systems to enable AI agents to play games of generic specification [26]. Cross-game agents must also consider how the personality and memory of the agent translates into new game contexts. To that end, research into *socially present agents* has explored how to control in-game behavior in reference to out-of-game context. Techniques explored include referring to historical interactions with players, simulating social roles common in a game, and explicitly signaling social and affective states not immediately relevant to in-game behavior [27]. Linking cross-game data on player-agent interactions to how social actions affect players will be crucial.

Cross-game agents may also serve as “game universe guides,” acting as a curator or tour guide to ensure players get the most out of a space of possible games. Important challenges involve understanding and eliciting player feedback on games, guiding players to new games as their interests shift, and sequencing the order of games played to optimize player experience. Cross-game data will be the linchpin to recommending different games and modeling how playing games in different orders (and ways) affects player experience.

### 5) How can a Game AI system generate cross-game content?

To date procedural content generation systems have been developed on a game-by-game basis. Cross-game content generation and adaptation explores how data acquired about a player in one game can improve content generation in another game in the ecosystem. Cross-game data enables

mining game designs for *general design knowledge*—a model mapping game mechanics to player behavior. Taken to its extreme, such design knowledge can lead to generating new games that span genres, moving beyond the current genre-focused efforts. Understanding cross-game player behavior can also allow recommending content from other games and ultimately lead to pre-adapting game experiences to players based on their behavior in other games.

### 6) How can a Game AI system add to the game ecosystem?

If AI as Producer focuses on an ecosystem of computer games, we might ask whether intelligent systems can automatically create new games that add to the ecosystem in meaningful ways. Computer game generation is a nascent area of Game AI research [28], [29], [30], [31]. Work to date has focused on a single genre at a time. Cross-game play puts a premium on new research to represent more generic game structures in a way that spans multiple genres. Many of the existing formalisms may be able to support such extensions, but how to best bridge genres remains an open question. More ambitious work will ultimately aspire to AI Producers that generate sets of games that complement one another, creating game ecosystems using a wealth of accumulated design knowledge.

Despite substantial efforts to reason on design knowledge, relatively little work has explored ways to acquire or refine general design knowledge. Current game industry efforts employ A/B testing for online game design—exemplified by Zynga’s practices. Analogous experimental methods have been used to understand how game designs impact player engagement and learning [32] or negative behavior [33]. Leveraging the unique potential of online distribution methods for *automated* rapid iteration and experimentation can open the way to addressing the challenges of extracting cross-game design knowledge. However, a number of research questions must first be addressed. First, an automated system must choose which designs to test, balancing benefits of testing against the high costs (in terms of player time, money, or negative reactions) of automated crowdsourced testing. Second, an automated system must be able to interpret the results of testing, including attributing credit/blame to aspects of design choices and appropriately changing designs in response. Third, an automated system must ultimately devise new design goals to explore when feedback indicates a given goal is unfeasible or unvaluable. Additionally, if user-generated content is available, a system might mine design principles from the content as a means of bootstrapping learning PCG systems.

## C. Community in Games

Game communities demand greater attention to how players impact one another’s experiences within a game, beyond the dynamics of cooperation and competition limited to a single session or round of play. Player communities extend in-game activities to a broader out-of-game social content. Careful *matchmaking* can group players for entertainment and engagement while *group-oriented content generation* can provide experiences tailored to sets of players. Further, communities themselves yield a wealth of user-generated content, posing opportunities to enhance interactions with PCG systems for better user- and system-generated content. However, with community comes a dark side: fraud, security violations, cheating, and abusive behavior. Game AI has an unique opportunity

to enhance negative behavior detection and automate responses beyond merely banning players.

*7) How can Game AI systems improve matchmaking and group content?* Matchmaking has traditionally focused on pairing players for competition to ensure even win rates. Online and social games, however, require grouping players for cooperative or synergistic goals. Beyond balancing win rates, players can be grouped to have complementary abilities or for purposes such as mentoring. Developing techniques to model player value as a social partner and using that information to create groups stand as key research challenges. Future developments will likely require adopting more sophisticated techniques from the social matching literature [34].

Group-oriented content generation extends PCG to the setting of engaging multiple players—with potentially conflicting interests—at once. Algorithms that balance the diverse needs of players in a given group, live game constraints (e.g. in terms of available players), and meet design goals are relevant research vectors. Both matchmaking and group-oriented content generation will require advances in modeling how players relate to one another socially and how these social and personal attributes interact with game content.

*8) How can a Game AI system reduce negative behavior?* Negative player behaviors in online games include fraud, security violations, cheating, and abusive behaviors. Fraudulent behavior commonly involves counterfeit game items sold for real-world money. Security violations compromise games through stealing players' accounts or financial data. Cheating spans hacking game systems to change their functionality to exploiting game bugs for personal gain or harm to others. Abusive behaviors include insulting other players or intentionally attempting to ruin their game experiences. Across these issues are a broad set of Game AI research challenges associated with responding appropriately to these behaviors. Note that many of these behaviors implicitly require cross-game agents—many forms of negative behavior manifest at the level of human players who act both within specific games and on game forums or other out-of-game socialization venues.

There is a wealth of research on fraud detection, security violations, and related challenges [35]. Little work, however, has addressed how Game AI agents should respond to these activities once detected. Game companies have only recently begun to systematically investigate ways to reduce negative behavior beyond banning players (e.g. [33]). We hypothesize that negative behavior can be reduced or mitigated by automatically adapting game content, rules, and mechanics to incentivize players toward pro-social behaviors. Game AI agents may minimize a player's negative impact by redirecting their actions away from others players. Detection may be improved by intentionally eliciting fraudulent or cheating behavior through an AI double-agent. Beyond negative reinforcement or punishment, Game AI agents can reward positive behavior or channel players toward more constructive pursuits.

*9) How can a Game AI system induce user-generated content?* User-generated content has become a major force for players' continuing interest in aging games. *Minecraft*, *Spore*, *Second Life*, and a host of other games have demonstrated the power of end-users to continually extend and enhance a live game. Despite the growing ubiquity of user-generated content

little research has developed methods to elicit needed content, interact with end-users to improve content, or mine design knowledge from this content.

Open questions for future user-generated content systems relate to how to best incorporate user content and feedback to improve the content created. Existing research has examined providing preference information (e.g. [36], [37]) or directly authoring the space of content (e.g. [29], [38], [39]). Enhancing user-generated content will require enabling users to teach PCG systems in new ways. *Interactive machine learning* has been developing techniques that optimize human abilities to train learning algorithms [40]. Integrating interactive machine learning approaches with user content creation interfaces can enable a new wave of learning PCG systems that improve through interaction with a player community. Key research problems include devising means to gather new modes of feedback, incorporating that feedback into PCG systems, and aggregating feedback from a diverse community of players.

#### D. Coupling the Real and Virtual Worlds

Games are increasingly coupled to the real world through input and output modalities that put a greater premium on a player's context. New input devices provide novel sensor information including GPS location (mobile devices), room layouts (Microsoft Kinect), motion data (Nintendo Wiimote, Playstation Move), sound (Kinect, webcams), and brain activity (NeuroSky, Emotiv). Output modalities have similarly become richer, including 3D displays (3D TV, Nintendo 3DS), second-screen experiences (Nintendo WiiU), virtual reality (Oculus Rift), augmented reality (mobile devices of all sorts), and potentially projection technologies. These technologies introduce nuances of player physical context including location, bodily motion data, and various physiological indicators. Beyond this information, games must also account for other aspects of player context including social circumstances or economic situation. Together this additional contextual information and output opportunities open research avenues related to incorporating real-world context into games, using out-of-game social network data in games and using games to proactively sense real-world information.

*10) How can a Game AI system utilize real-world context within a game?* Real-world data opens many avenues for game experiences that overlay on real-world settings or leverage real-world semantic information for new forms of gameplay. Human-computer interaction research has a rich literature on addressing the challenges and nuances of context, but has seen little adoption in the context of Game AI [41]. Understanding how AI agents can (or should) use context is an open problem for future Game AI agents.

Real-world settings enable new game types including augmented reality games and alternate reality games. *Augmented reality games* visually project virtual game content onto the real-world situation. *Alternate reality games* overlay fictional contexts on real-world settings without necessarily requiring a visual overlay—Google's *Ingress* is a prominent example. Both kinds of games, however, are currently circumscribed by challenges in authoring new content and fitting it to new, unanticipated real world contexts—a promising avenue for future PCG research. Preliminary work on merging the virtual

and real has explored dynamically adapting alternate reality game quests to new physical locations [42], [43]. The next stages of this work may use add other real world context in the game, such as social or economic context.

An alternative perspective on coupling real and virtual worlds involves using the semantic information present in real-world data. *Data Games* engage players in understanding semantic information in open data sets (e.g. US census data) through automatically mapping data into game content [44]. Future work should explore how in-game agents can leverage out-of-game data for richer interaction with players and tighter coupling of game worlds and real-world contexts.

**11) How can a Game AI system leverage out-of-game social networks?** Modeling in-game social interactions, social networks, and player communities can enable Game AI agents to interact with players as part of a social world that (partially) overlaps the in-game world. Current research on online game social networks has explored detecting social information including group identities [45], shared housing networks [46], and real-money trade [47]. Outside of games a wealth of prior work on modeling techniques for social and economic networks exists [48]. Here we ask: how can Game AI systems use social networks from outside a game to improve in-game experiences?

Overall, relating in-game interactions to out-of-game social interaction remains underexplored. Open research problems include uncovering how network structures can be used to draw users into a company’s game ecosystem or how to encourage users to move among games. Out-of-game social networks such as *Twitter* and *Facebook* additionally contain a wealth of user-generated material that reveals sentiments, preferences, attitudes, and non-game product usage. We speculate this data could be used to influence motivational content generation and integrate ads into game content in non-trivial ways.

**12) Can an intelligent system use games to “proactively sense” the real world?** In the age of big data, companies routinely collect data about users to improving products or otherwise monetizing user behavior. Yet, many aspects of player motivations or real-world context remain hidden from these efforts. *Proactive sensing* is the use of game content to elicit data about the real-world that might not otherwise be collected by passive observation of users. For example, consider attempting to learn whether a new coffee shop has good coffee. Game content—possibly a quest in an alternate reality game—can be generated that requires players within proximity of the coffee shop to visit and rate the shop, thus generating new data that might not otherwise have been generated. Additional applications of the proactive sensing paradigm can target other hidden real-world information such as player preferences, skills, and attitudes or semantic information about real-world objects and locations.

Proactive sensing research will require modeling the quality of information known about the world player abilities to provide that information, and challenges in generating game-relevant content and agent behaviors to elicit this information. Human computation research has modeled human abilities to perform tasks that provide computers with information about the real world [49]. Extending these approaches to games will involve incorporating player motivation to perform or

complete game tasks. Key research problems will include balancing between game design goals and information needs and generating game tasks and agent behaviors appropriate to a wide variety of information needs.

## V. CONCLUSIONS

The digital game industry is experiencing a shift to persistent games, business models emphasizing game ecosystems, more support for game communities, and new considerations for incorporating real world context into game worlds. We see these advances as positive signs of growth and maturity in the game industry. We also see these advances pushing the bounds on the scalability of game development practices.

Artificial intelligence, computational intelligence, and machine learning have always excelled at addressing problems of scalability by automating tasks and dynamically adapting system behavior. Game AI has always been an integral part of computer game development. AI Actors have enhanced player experiences by supporting players’ suspension of disbelief and dynamically managing dramatic contexts. AI Designers have supported and augmented the development of individual games through procedural content generation. We envision AI Producers taking on a new role of augmenting and scaling the game production pipeline, supporting the entire span of live operations in games, enhancing cross-game interoperations, nurturing strong player communities, and coupling real and virtual contexts. This vision is bolstered by the massive amounts of data being generated and collected by the game development industry.

The twelve new Game AI research questions here require game developers to see Game AI as part of live game production. AI for game production does not supplant previous challenges in Game AI—it extends the scope of the field of Game AI as a whole. Addressing these problems with credible AI solutions will result in immediate relevance to game developers and may present a new vector for industry game development and academic Game AI research collaboration.

## REFERENCES

- [1] J. Laird and M. VanLent, “Human-level AI’s killer application: Interactive computer games,” *AI magazine*, vol. 22, no. 2, p. 15, 2001.
- [2] M. Mateas, “An Oz-centric review of interactive drama and believable agents,” in *Artificial Intelligence Today*, ser. Lecture Notes in Computer Science, M. J. Wooldridge and M. Veloso, Eds. Springer Berlin Heidelberg, 1999, vol. 1600, pp. 297–328.
- [3] D. L. Roberts and C. L. Isbell, “A survey and qualitative analysis of recent advances in drama management,” *International Transactions on Systems Science and Applications, Special Issue on Agent Based Systems for Human Learning*, vol. 4, no. 2, pp. 61–75, 2008.
- [4] M. O. Riedl and V. Bulitko, “Interactive narrative: An intelligent systems approach,” *AI Magazine*, 2013.
- [5] M. Hendrikx, S. Meijer, J. Van Der Velden, and A. Iosup, “Procedural content generation for games: A survey,” *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 9, no. 1, p. 1, 2013.
- [6] J. Togelius, G. Yannakakis, K. Stanley, and C. Browne, “Search-based procedural content generation: A taxonomy and survey,” *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 3, no. 3, pp. 172–186, 2011.
- [7] G. N. Yannakakis, “Game AI revisited,” in *Computing Frontiers*, 2012, pp. 285–292.

- [8] A. M. Smith, E. Andersen, M. Mateas, and Z. Popović, “A case study of expressively constrainable level design automation tools for a puzzle game,” in *7th International Conference on the Foundations of Digital Games*, 2012, pp. 156–163.
- [9] E. Andersen, S. Gulwani, and Z. Popovic, “A trace-based framework for analyzing and synthesizing educational progressions,” in *ACM SIGCHI Conference on Human Factors in Computing Systems*, 2013.
- [10] H. Yu and M. O. Riedl, “A sequential recommendation approach for interactive personalized story generation,” in *11th International Conference on Autonomous Agents and Multiagent Systems*, 2012, pp. 71–78.
- [11] D. Thue, V. Bulitko, M. Spetch, and E. Wasylshen, “Interactive storytelling: A player modelling approach,” in *3rd AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2007.
- [12] N. Shaker, G. N. Yannakakis, and J. Togelius, “Towards automatic personalized content generation for platform games,” in *6th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2010.
- [13] M. Seif El-Nasr, A. Drachen, and A. Canossa, Eds., *Game Analytics*. Springer London, 2013.
- [14] A. Smith, C. Lewis, K. Hullett, G. Smith, and A. Sullivan, “An inclusive view of player modeling,” in *6th International Conference on the Foundations of Digital Games*, 2011.
- [15] B. Harrison and D. L. Roberts, “Using sequential observations to model and predict player behavior,” in *6th International Conference on the Foundations of Digital Games*, 2011.
- [16] B. G. Weber, M. Mateas, and A. H. Jhala, “Using data mining to model player experience,” in *FDG Workshop on Evaluating Player Experience*, 2011.
- [17] G. Yannakakis and J. Togelius, “Experience-driven procedural content generation,” *IEEE Transactions on Affective Computing*, vol. 2, no. 3, pp. 147–161, 2011.
- [18] T. Bickmore and D. Schulman, “A virtual laboratory for studying long-term relationships between humans and virtual agents,” in *8th International Conference on Autonomous Agents and Multiagent Systems*, 2009, pp. 297–304.
- [19] D. L. Silver, Q. Yang, and L. Li, “Lifelong machine learning systems: Beyond learning algorithms,” in *AAAI Spring Symposium Series*, 2013.
- [20] K. VanLehn, “The behavior of tutoring systems,” *International Journal of Artificial Intelligence in Education*, vol. 16, no. 3, pp. 227–265, 2006.
- [21] R. Hunicke and V. Chapman, “AI for dynamic difficulty adjustment in games,” in *AAAI Workshop on Challenges in Game Artificial Intelligence*, 2004.
- [22] B. Magerko, B. Stensrud, and L. Holt, “Bringing the schoolhouse inside the box - a tool for engaging, individualized training,” in *25th Army Science Conference*, 2006.
- [23] H. Yu and T. Trawick, “Personalized procedural content generation to minimize frustration and boredom based on ranking algorithm,” in *7th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2011.
- [24] A. Zook and M. O. Riedl, “A temporal data-driven player model for dynamic difficulty adjustment,” in *8th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2012.
- [25] B. Li, S. Lee-Urban, D. S. Appling, and M. O. Riedl, “Crowdsourcing narrative intelligence,” *Advances in Cognitive Systems*, vol. 2, pp. 25–42, 2012.
- [26] M. Genesereth, N. Love, and B. Pell, “General game playing: Overview of the AAAI competition,” *AI magazine*, vol. 26, no. 2, p. 62, 2005.
- [27] A. Pereira, R. Prada, and A. Paiva, “Socially present board game opponents,” in *Advances in Computer Entertainment*. Springer, 2012, pp. 101–116.
- [28] J. Togelius and J. Schmidhuber, “An experiment in automatic game design,” in *IEEE Symposium on Computational Intelligence and Games*, 2008, pp. 111–118.
- [29] A. M. Smith and M. Mateas, “Variations Forever: Flexibly generating rulesets from a sculptable design space of mini-games,” in *IEEE Conference on Computational Intelligence and Games*, 2010.
- [30] M. Cook, S. Colton, and J. Gow, “Initial results from co-operative co-evolution for automated platformer design,” in *EvoGames 2012*, 2012.
- [31] K. Hartsook, A. Zook, S. Das, and M. Riedl, “Toward supporting storytellers with procedurally generated game worlds,” in *IEEE Conference on Computational Intelligence in Games*, 2011.
- [32] D. Lomas, K. Patel, J. L. Forlizzi, and K. R. Koedinger, “Optimizing challenge in an educational game using large-scale design experiments,” in *ACM SIGCHI Conference on Human Factors in Computing Systems*, 2013, pp. 89–98.
- [33] J. Lin, “The science behind shaping player behavior in online games,” 2013, game Developers Conference. [Online]. Available: <http://www.gdcvault.com/play/1017940/The-Science-Behind-Shaping-Player>
- [34] L. Terveen and D. W. McDonald, “Social matching: A framework and research agenda,” *ACM Transactions on Computer-Human Interaction*, vol. 12, no. 3, pp. 401–434, 2005.
- [35] C. Phua, V. C. S. Lee, K. Smith-Miles, and R. W. Gayler, “A comprehensive survey of data mining-based fraud detection research,” *Computing Research Repository*, vol. abs/1009.6119, 2010.
- [36] E. Hastings, R. K. Guha, and K. Stanley, “Automatic content generation in the galactic arms race video game,” *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 1, pp. 245–263, 2009.
- [37] S. Risi, J. Lehman, D. B. D’Ambrosio, R. Hall, and K. O. Stanley, “Combining search-based procedural content generation and social gaming in the petalz video game,” in *8th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2012.
- [38] G. Smith, J. Whitehead, and M. Mateas, “Tanagra: Reactive planning and constraint solving for mixed-initiative level design,” *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 3, no. 3, pp. 201–215, 2011.
- [39] J. Dormans, “Machinations: Elemental feedback structures for game design,” in *GAMEON-NA*, 2009, pp. 33–40.
- [40] S. Amershi, J. Fogarty, A. Kapoor, and D. Tan, “Effective end-user interaction with machine learning,” in *25th AAAI Conference on Artificial Intelligence*, 2011, pp. 7–11.
- [41] J.-y. Hong, E.-h. Suh, and S.-J. Kim, “Context-aware systems: A literature review and classification,” *Expert Systems with Applications*, vol. 36, no. 4, pp. 8509–8522, 2009.
- [42] A. Macvean, S. Hajarnis, B. Headrick, A. Ferguson, C. Barve, D. Karnik, and M. O. Riedl, “WeQuest: Scalable alternate reality games through end-user content authoring,” in *8th International Conference on Advances in Computer Entertainment Technology*, 2011, p. 22.
- [43] A. Gustafsson, J. Bichard, L. Brunnberg, O. Juhlin, and M. Combetto, “Believable environments: generating interactive storytelling in vast location-based pervasive games,” in *ACM SIGCHI International Conference on Advances in Computer Entertainment Technology*, 2006, p. 24.
- [44] M. G. Friberger, J. Togelius, A. B. Cardona, M. Ermacora, A. Moustén, M. Jensen, V. Tanase, and U. Brøndsted, “Data games,” in *4th Workshop on Procedural Content Generation*, 2013.
- [45] C. Grappiolo, J. Togelius, and G. N. Yannakakis, “Interaction-based group identity detection via reinforcement learning and artificial evolution,” in *GECCO Evolutionary Computation and Multi-agent Systems and Simulation Workshop*, 2013.
- [46] M. A. Ahmad, B. Keegan, D. Williams, J. Srivastava, and N. Contractor, “Trust amongst rogues? a hypergraph approach for comparing clandestine trust networks in MMOGs,” in *Fifth International AAAI Conference on Weblogs and Social Media*, 2011, pp. 17–21.
- [47] A. Fujita, H. Itsuki, and H. Matsubara, “Detecting real money traders in MMORPG by using trading network,” in *7th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2011, pp. 26–31.
- [48] M. O. Jackson, *Social and economic networks*. Princeton University Press, 2010.
- [49] E. Law and L. Ahn, “Human computation,” *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 5, no. 3, pp. 1–121, 2011.

# Data-Driven Approaches to Game Player Modeling: A Systematic Literature Review

DANIAL HOOSHYAR, MOSLEM YOUSEFI, and HEUISEOK LIM, Korea University

Modeling and predicting player behavior is of the utmost importance in developing games. Experience has proven that, while theory-driven approaches are able to comprehend and justify a model's choices, such models frequently fail to encompass necessary features because of a lack of insight of the model builders. In contrast, data-driven approaches rely much less on expertise, and thus offer certain potential advantages. Hence, this study conducts a systematic review of the extant research on data-driven approaches to game player modeling. To this end, we have assessed experimental studies of such approaches over a nine-year period, from 2008 to 2016; this survey yielded 46 research studies of significance. We found that these studies pertained to three main areas of focus concerning the uses of data-driven approaches in game player modeling. One research area involved the objectives of data-driven approaches in game player modeling: behavior modeling and goal recognition. Another concerned methods: classification, clustering, regression, and evolutionary algorithm. The third was comprised of the current challenges and promising research directions for data-driven approaches in game player modeling.

CCS Concepts: • **Information systems** → **Massively multiplayer online games**; *Data mining*; • **Computing methodologies** → **Machine learning**; *Artificial intelligence*; • **Applied computing** → **Computer games**;

Additional Key Words and Phrases: Game player modeling, data-driven approaches, computational models, systematic literature review (SLR)

## ACM Reference format:

Danial Hooshyar, Moslem Yousefi, and Heuiseok Lim. 2018. Data-Driven Approaches to Game Player Modeling: A Systematic Literature Review. *ACM Comput. Surv.* 50, 6, Article 90 (January 2018), 19 pages.

<https://doi.org/10.1145/3145814>

90

## 1 INTRODUCTION

Player modeling entails describing a game player's traits and tendencies within a model. Such characteristics may include a player's actions and behavior, dispositions and style, motivations, and aims (Bakkes et al. 2012; Yannakakis et al. 2013). These models can allow a game to tailor its content or goals automatically to suit the needs of a specific player. Adaptation does not necessarily require modeling, since games can respond merely to alterations in the world of play, or to a player's biometric data (Cowley et al. 2016). Nevertheless, there are multiple advantages to developing a

---

This work is supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No. 2016R1A2B2015912).

Author's addresses: D. Hooshyar and H. Lim, Lyceum, Department of Computer Science and Engineering, Korea University, Anam-ro, Seongbuk-gu, Seoul, the Republic of Korea; emails: danial.hooshyar@gmail.com, limhseok@korea.ac.kr; M. Yousefi, School of Civil, Environmental and Architectural Engineering, Korea University, Anam-ro, Seongbuk-gu, Seoul, the Republic of Korea; email: yousefi.moslem@gmail.com.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2018 ACM 0360-0300/2018/01-ART90 \$15.00

<https://doi.org/10.1145/3145814>

player model to mediate these responses. First, a player model offers comprehension of a player's behavior, and thus can justify certain adaptations. Second, it makes adaptations generalizable to other games.

There are three domains of data conducive to the building of player models. First among these is gameplay data, that is, data collected immediately from interactions between player and game within the game world. Second is subjective data—that is, data collected via questionnaires (involving, for instance, demographics, emotional states, personality tests, or psychometrics). Objective data constitute a third domain, culled from biometrical observations. There are two basic approaches that are employed in building player models, one theory-driven and one data-driven. A theory-driven approach relies on the social sciences, particularly experimental psychology; it entails advancing a model derived from literature and domain experience, then subsequently using empirical methods to validate the model (Lucas et al. 2013). A data-driven approach, in contrast, relies on the natural sciences and computer science; it first gathers a significant quantity of measurements before applying computational methods to derive models, either fully or semi-automatically (Yannakakis et al. 2013).

Experience has proven that, while theory-driven approaches are able to comprehend and justify a model's choices, such models are frequently missing relevant features because their architects lack sufficient insight. Games, as inherently open-ended constructs, tend to create a massive space of actions. For this reason, data-driven approaches show promise because they do not rely on expert domain knowledge. In addition to being less dependent on expert guidance, data-driven approaches can likewise collect huge quantities of low-level user actions prior to any modular description of those actions (Min et al. 2014). Furthermore, this allows for the granular examination of game player behavior, leading to further understanding. Player movement errors can be examined, for instance, in order to gain insight into the misconceptions that caused them. Such knowledge can then generate more sophisticated cognitive models and a more comprehensive understanding of user behavior.

For these reasons, we are convinced of the promise of data-driven approaches to game player modeling, a belief shared by a number of works (Cowley et al. 2009; Galway et al. 2008; Lee et al. 2014; Machado et al. 2011; Yannakakis 2012; Zook et al. 2012). However, the increasing interest in this field has yet to lead to any effort to link its concepts and techniques. Such an organization of knowledge is vitally important, and requires a systematic review of the current empirical evidence regarding the benefits, challenges, and application of data-driven approaches to game player modeling. To our knowledge, to date there has been neither such a review nor even a summary of empirical evidence. In executing a systematic review of the empirical evidence, this article aims to make several contributions. First, it seeks to offer a review, both systematic and comprehensive, detailing how data-driven approaches have been implemented in game player modeling. Second, it intends to provide a classification of the assorted data-driven approaches to game player modeling. In so doing, it will (third) offer an understanding of the current challenges and promising future directions in the field.

The structure of the article is as follows: Section 2 details the methodology of our review; Section 3 outlines the results and analyzes important studies; Section 4 elaborates on the discussion and considers future directions, limitations, and conclusions.

## 2 RESEARCH METHOD

A systematic literature review (SLR) necessitates a comprehensive and impartial search plan, so as to ensure the completeness of the search of materials under consideration. The recent upswing in interest in the field of data-driven approaches to game player modeling has yet to produce

any such comprehensive effort to survey its concepts, methods, and problems; our aim here is to employ Keele's methodology in a much-needed SLR (Keele 2007).

## 2.1 Research Questions

In this SLR, our primary research question is as follows: "What is the state-of-the-art in applying data-driven approaches in game player modeling?" We break this overarching question into three subordinate questions: the first two we address in the results section, because they are empirical in nature, and save the third (more speculative) question for the discussion section.

- RQ1: Considering the surveyed research on data-driven approaches to game player modeling, what basic objectives and methods do the researchers employ?
- RQ2: What are the challenges of using data-driven approaches in game player modeling?
- RQ3: Looking forward, what are promising future directions in data-driven game player modeling?

## 2.2 Database and Keywords

We classified five databases in our search: Springer Link, Science Direct, DBLP, IEEE Xplore, and ACM Digital Library. Additionally, we searched sources outside those databases, in particular scanning bibliographies of relevant articles. However, we did not include Google Scholar among these sources, on account of both the poor precision of its results and the extent to which it overlapped with other data sources. In order to capture the entirety of extant research in this area, we employed a grouping of various keywords. The following search terms were used together ("data-driven approaches" OR "player modeling" OR "data mining" OR "learning individual behavior" OR "user modeling") AND ("game").

## 2.3 Search Procedure

We display in Figure 1 the three stages of study selection we employed in our systematic review. Throughout these stages, we held meetings to arrive at a consensus concerning any disagreements that arose amongst the reviewers during the selection, assessment, or data extraction processes. We sought thereby to reduce bias to a minimum, and likewise to boost confidence in the study selection process.

As Table 1 shows, we employed criteria of inclusion upon the entirety of the studies gathered in the multiple phases of selection; in doing so, we sought to mine the electronic databases for studies potentially pertinent to the research questions of our SLR. In order to limit the sample to current studies, we established 2008 and 2016 as temporal parameters for the search. There was, in point of fact, a significant quantity of research concerning the use of data-driven methods in game player modeling that predated 2008. We settled on that cutoff, however, because it marks a turning point in the expansion of the interest in, and publications about, the use of data-driven approaches in games. It was immediately followed by the inauguration and development of a number of influential yearly meetings: the IEEE Conference on Computational Intelligence and Games (CIG), the AAAI Artificial Intelligence, and Interactive Digital Entertainment (AIIDE) conference series. For publications that fell within the temporal parameters for our literature search, all titles were screened separately by three reviewers in order to remove publications that obviously fell outside the interests of the current study (such as those concerning approaches to game-based learning and gamification). The next round of evaluation focused on abstracts: the three reviewers examined the abstracts of 238 articles to determine their eligibility, removing all but 94 of them. In the case of any uncertainty concerning a study's relevance based on its title or abstract, we chose to include the study in our full-text evaluation.

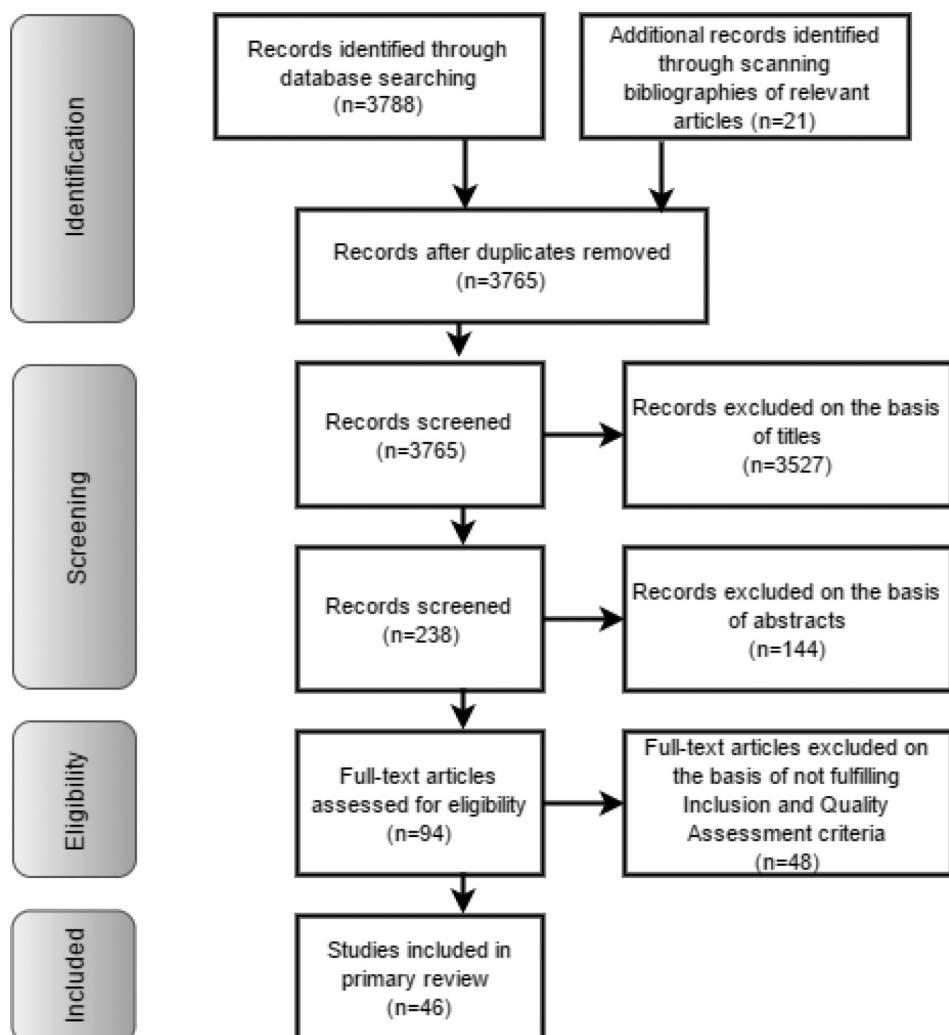


Fig. 1. Study selection procedure.

Table 1. Selection Criteria for Inclusion of Papers

| Inclusion criteria |   |
|--------------------|---|
| IC1                | Peer-reviewed articles published in journals or full-length articles published in International Conference/Workshop Proceedings |
| IC2                | Presenting full results   |
| IC3                | Dated between 2008 and 2016   |
| IC4                | English language studies  |

Third, the full texts of these 94 articles were screened for final eligibility, with matters of inclusion weighed according to the inclusion criteria and quality attributes of Table 2. The 94 articles were distributed to the three reviewers, who carefully reviewed each one in an effort to correct for any bias. The researchers applied the four criteria to assess quality of the articles that had

Table 2. Quality Appraisal Criteria for SLR

| Quality appraisal criteria |  |
|----------------------------|--|
| QC1                        | Frequency of citations   |
| QC2                        | Methodological contributions   |
| QC3                        | Sound methodology: Clearly stated goals, limitations, and contributions with transparent and explained experimentation |
| QC4                        | Sufficient presentation of the findings  |

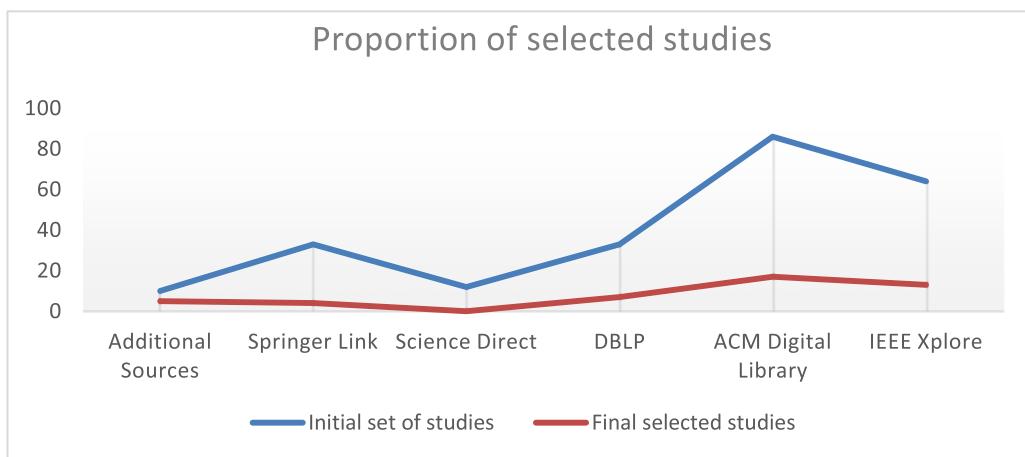


Fig. 2. Ratio of selected studies.

been filtered through the second phase. They used a scale of 1 to 5 (with 5 denoting “very high,” 4 “high,” 3 “medium,” 2 “low,” and 1 “very low” on the given category); total scores thus could span from 20 (highest quality, without much risk of bias) to 4 (poor quality, potentially significant bias). Any conflicting opinions arising between the reviewers in the process of selection, assessment, and data extraction were addressed in meetings to reach a consensus. The result of this exhaustive screening process was to winnow the 94 full-text research articles down to 46 research studies of the application of data-driven approaches to game player modeling that were conducted from 2008 to 2016. Thus, the suitability of the articles finally selected for inclusion was assured. Figure 2 presents the number of articles initially derived from each source in proportion to the articles included in the final tally.

The reviewers carefully evaluated the full text of all the studies that passed the second phase, and then catalogued relevant material using a pre-established form for the extraction of data according to research strategy used (Keele 2007). The form included *research discipline*, *data gathering*, *research objectives*, *data acquisition*, *analysis technique*, and *results*. The demographics and overview of these studies are presented in Appendix A.

### 3 RESULTS AND KEY STUDIES ANALYSIS

#### 3.1 Research Discipline

While specific topics vary according to study, the majority address three primary areas: behavior/experience modeling, goal/plan recognition, and procedural content generation. This list of areas of potential is by no means comprehensive, but it represents a major slice of research and development at present.

### 3.2 Data Gathering

The reviewed research studies gathered data from different data sources (games). More details, including game purposes, are given in [Table 3](#). In total, 32 concerned entertainment and 14 addressed learning-focused games. While entertainment and learning frequently overlap (since many games aim at both), we divided them along these lines to distinguish specifically pedagogical games from the rest.

### 3.3 Research Question 1

**RQ1:** Considering the surveyed research on data-driven approaches to game player modeling, what basic objectives and methods do the researchers employ?

In our examination of the reviewed works, we found that most studies share a basic set of objectives: player behavior modeling/experience modeling, goal/plan recognition, and procedural content generation. Authors have used a range of Artificial Intelligence (AI) methods to pursue some specific tasks. This section presents the fundamental objectives, approaches employed, and their application in some works. [Table 4](#) classifies and presents the selected studies according to research objectives. We characterized as “behavioral modeling” methods of player behavior modeling, experience modeling, and procedural content generation present in the reviewed studies, because they frequently necessitate a functional model of the player and/or his behavior. Even though player modeling significantly overlaps with goal/plan recognition, as such recognition entails a problem in which action sequences are interpreted in terms of the goal they are most probably attempting to realize, we decoupled behavior modeling from plan/goal recognition because it sidesteps the process of predicting goals and deducing behavior from them and instead predicts behavior directly. In the pool of studies under review, 34 address player behavior modeling, while 12 address goal recognition. As this distribution clearly demonstrates, the field has been tilted towards player behavior modeling, such that this area dominates the spotlight with regards to current research and advances.

Our review also considered another vital parameter: namely, the method of data analysis which researchers apply to their collected data (see [Table 5](#)). As [Table 5](#) clearly shows, classification is strongly favored in player modeling, whereas clustering and regression are employed in a more limited fashion. The area of player modeling appears moreover to contain a particularly varied and rich array of AI methods. In order to delineate the classification of methods presented here, it should be understood that supervised learning indicates learning a model in which instances of datasets are aligned with target values (such as classes); target values are thus a prerequisite for supervised learning (as in neural networks). In contrast, unsupervised learning describes algorithms that locate patterns in datasets without target values (e.g., k-means). Evolutionary computation indicates population-based global stochastic optimization algorithms, such as genetic algorithms. One might of course categorize these methods in a different manner, but our contention is that the classification we put forward is succinct and adheres to typical method classifications in artificial and computational intelligence.

**3.3.1 Behavior Modeling.** As [Table 4](#) shows, one of the main objectives of the reviewed research studies is to detect, identify, and model the behavior of players, as behavior modeling constitutes the research objective of 34 studies.

In particular, these studies aim to model the behavior and experience of game players, and from that to design-predictive models of user behavior and movement to generate game contents (see A1-A5, A7, A12, A14-A23, A27-A29, A31, A33-A40, and A42-A46 in Appendix A). Several AI techniques have been studied and used for modeling player behavior by the authors of reviewed research studies, such as classification, clustering, regression and evolutionary algorithms. We

Table 3. Games Used in the Reviewed Research Studies

| Paper ID in Appendix A     | Game  | Purpose       | Reference   |
|----------------------------|---|---------------|---|
| A1                         | Alice in AreaLand   | learning      | (Falakmasir et al. 2016)  |
| A2                         | Combat with Monsters  | learning      | (Zook et al. 2012)  |
| A3                         | The Hunter  | entertainment | (Ramirez-Cano et al. 2010)  |
| A4                         | Forza Motorsport 5  | entertainment | (Harpstead et al. 2015)   |
| A5                         | World of Warcraft   | entertainment | (Harrison and Roberts 2011)   |
| A6,<br>A13,A24,A25,<br>A41 | CRYSTAL ISLAND:<br>OUTBREAK/<br>CRYSTAL ISLAND:<br>UNCHARTED<br>DISCOVERY | learning      | (Baikadi et al. 2011; Ha et al. 2012;<br>Min et al. 2016a, 2016b, 2014)                         |
| A7, A17                    | Infinite Mario  | entertainment | (Shaker et al. 2010; Weber et al. 2011)   |
| A8,A26,A30,<br>A32         | StarCraft   | entertainment | (Bisson et al. 2015; Kabanza et al. 2010;<br>Synnaeve and Bessiere 2011; Weber and Mateas 2009) |
| A9                         | The Legend of Zelda   | entertainment | (Gold 2010)   |
| A10                        | Rush Football   | entertainment | (Laviers and Sukthankar 2011)   |
|                            |   |               |   |
| A11                        | The Restaurant Game   | entertainment | (Orkin et al. 2010)   |
| A12,A29                    | Tomb Raider:<br>Underworld  | entertainment | (Drachen et al. 2009;<br>Mahlmann et al. 2010)  |
| A14,A38                    | Pac-Man   | entertainment | (Cowley et al. 2009; 2013)  |
| A15                        | Anchorhead  | entertainment | (Sharma et al. 2010)  |
| A16                        | Blindmaze   | entertainment | (Etheredge et al. 2013)   |
| A18                        | Civilization IV   | entertainment | (Spronck and den Teuling 2010)  |
| A19                        | Cannon & spaceship  | entertainment | (Missura and Gärtner 2009)  |
| A20                        | Bug Smasher   | entertainment | (Yannakakis and Hallam 2009)  |
| A21                        | DragonBox   | learning      | (Lee et al. 2014)   |
| A22,A42                    | REFRACTION  | learning      | (Butler et al. 2015; Liu et al. 2013)   |
| A23                        | Choose-Your-Own-Adventure   | entertainment | (Yu and Riedl 2012)   |
| A27                        | Sony Everquest II   | entertainment | (Borbora and Srivastava 2012)   |
| A28                        | Super Mario Bros  | entertainment | (Pedersen et al. 2010)  |
| A31                        | Madden NFL 11   | entertainment | (Weber et al. 2011)   |
| A33                        | Snakeotron  | entertainment | (Gow et al. 2012)   |
| A34                        | Food Distribution   | learning      | (Luo et al. 2016)   |
| A35                        | Space Invaders  | entertainment | (Anagnostou and Maragoudakis 2009)  |
| A36                        | Destiny   | entertainment | (Tamassia et al. 2016)  |
| A37                        | I Am Playr/Lyroke   | entertainment | (Xie et al. 2014)   |
| A39                        | SeaGame   | learning      | (Bellotti et al. 2009)  |
| A40                        | Solving the Incognitum  | learning      | (Valls-Vargas et al. 2015)  |
| A43                        | Resource Management Game  | learning      | (Grappiolo et al. 2011)   |
| A44                        | Battle with monsters  | entertainment | (Zook and Riedl 2012)   |
| A45                        | SUPER MONKEY BALL 2   | entertainment | (Cowley et al. 2014)  |
| A46                        | Table tennis game   | entertainment | (Gao et al. 2016)   |

Table 4. Research Objectives of the Reviewed Studies

| Research objectives (goals)  | Number of papers | Reference  |
|--|------------------|--|
| Player behavior modeling/<br>Experience modeling/Procedural content generation | 34               | (Anagnostou and Maragoudakis 2009; Bellotti et al. 2009; Borbora and Srivastava 2012, Butler et al. 2015; Cowley et al. 2009, 2013, 2014; Drachen et al. 2009; Etheredge et al. 2013; Falakmasir et al. 2016; Gao et al. 2016; Gow et al. 2012; Grappiolo et al. 2011; Harpstead et al. 2015; Harrison and Roberts 2011; Lee et al. 2014; Liu et al. 2013; Luo et al. 2016; Mahlmann et al. 2010; Missura and Gärtnner 2009; Pedersen et al. 2010, Ramirez-Cano et al. 2010; Shaker et al. 2010; Sharma et al. 2010; Spronck and den Teuling 2010; Tamassia et al. 2016; Valls-Vargas et al. 2015; Weber et al. 2011a, 2011b; Xie et al. 2014; Yannakakis and Hallam 2009; Yu and Riedl 2012; Zook et al. 2012; Zook and Riedl 2012) |
| Goal/plan recognition  | 12               | (Baikadi et al. 2011; Bisson et al. 2015; Gold 2010; Ha et al. 2012; Kabanza et al. 2010; Laviars and Sukthankar 2011; Min et al. 2014, 2016a, 2016b, 2014; Orkin et al. 2010; Synnaeve and Bessiere 2011; Weber and Mateas 2009)  |

characterize the AI methods that are primary or secondary in each approach. Primary methods denote the techniques most frequently applied in the literature, whereas secondary methods denote ones that appear in a significant number (but not a majority) of studies.

In reviewing the studies given in Table 5, we discovered that a number of classification algorithms are employed in the behavioral modeling of game players. These include Hidden Markov Models (HMM), Bayesian Networks (BN), Markov Logic Networks (MLN), Support Vector Machines (SVM), Neural Networks (NN), Long Short-Term Memory (LSTM), Recursive Neural Networks (RNN), K-Nearest Neighbor (KNN), Decision Trees (DT), and Deep Learning (DL). The classification algorithms employed most often for behavior modeling were the NN and its variant, as well as MLN and HMM, which thus garnered increased attention concerning their modeling potential (see A16, A17, A20-A22, A28, A34, A36 and A43 in Appendix A). After these, the next in popularity were SVM and DT (A19, A37, A38 and A40). Note that classification methods were sometimes paired with clustering or evolutionary algorithms in the studies under review; in such

Table 5. Data Analysis Method of the Reviewed Studies

| Data analysis method    | Algorithm  | Category                 | Number of papers | Reference   |
|-------------------------|--|--------------------------|------------------|---|
| Classification          | Hidden Markov Models, Bayesian Network, Markov Logic Networks, Input-Output Hidden Markov Model, Decision Tree, Support Vector Machine, Neural Network, Recursive Neural networks, Long Short-Term Memory Network, Deep learning, K-Nearest Neighbor | Supervised Learning      | 31               | (Balkadi et al. 2011; Bisson et al. 2015; Butler et al. 2015; Cowley et al. 2009; Cowley et al. 2013; Etheredge et al. 2013; Gold 2010; Grappiolo et al. 2011; Ha et al. 2012; Harrison and Roberts 2011; Kabanza et al. 2010; Laviers and Sukthankar 2011; Lee et al. 2014; Liu et al. 2013; Luo et al. 2016; Mahlmann et al. 2010; Min et al. 2014; 2016a; 2016b; Missura and Gärtner 2009; Orkin et al. 2010; Pedersen et al. 2010; Shaker et al. 2010; Spronck and den Teuling 2010; Synnaeve and Bessiere 2011; Tamassia et al. 2016; Valls-Vargas et al. 2015; Xie et al. 2014; Yannakakis and Hallam 2009; Yu and Riedl 2012; Zook and Riedl 2012) |
| Clustering              | k-means, CURE (Clustering Using REpresentatives), SOM (Self-Organizing Map), Spectral clustering, LDA (Linear Discriminant Analysis)   | Unsupervised Learning    | 7                | (Anagnostou and Maragoudakis 2009; Borbora and Srivastava 2012; Cowley et al. 2014; Drachen et al. 2009; Gow et al. 2012; Ramirez-Cano et al. 2010; Sharma et al. 2010)   |
| Regression              | Linear regression, Regression trees, Additive regression, Logistic regression  | Supervised Learning      | 6                | (Falakmasir et al. 2016; Gao et al. 2016; Harpstead et al. 2015; Weber and Mateas 2009; Weber et al. 2011a, 2011b)  |
| Evolutionary Algorithms | Genetic algorithm  | Evolutionary computation | 2                | (Bellotti et al. 2009; Zook et al. 2012)  |

cases, we labeled them under classification, since the data-driven methods those articles advanced depend primarily on the methods of classification.

Liu et al. (2013), for instance, worked out a method to predict player behavior and movement in an educational game. Their algorithm selects from a combination of methods—Markov models, state aggregation, and player heuristic search—according to whichever offers the largest amount of data. Their approach promises to minimize the burden of hand-designing a cognitive model and system-specific features. Lee et al. (2014) likewise developed a framework for predicting player movements within a pair of puzzle games, which, they showed, greatly outperformed both games' baselines. They proposed a data-driven approach, a mixture of a one-depth heuristic search model and a data-driven Markov model with no knowledge of individual history other than the current game state, to learning individual behavior. They added to the prior framework by constructing a state-action graph and employing methods of feature selection to minimize the number of features for each state. They then applied this new framework to another game (DragonBox) to ascertain its applicability for extension, and reported similar success. The studies undertaken by Lee et al. (2014) and Liu et al. (2013) hold the potential to address the two main issues of purely data-driven approaches—their failure to offer any semantically meaningful interpretation of outputs, on the one hand, and their shortcomings in algorithmic efficiency on the other. However, unlike the method proposed by Lee et al. (2014), Liu et al. (2013) assume that players do not change over

time and define a user reward as a combination of pre-defined heuristics. Thus, the quality of the learned policy is strongly dependent on these heuristics, which are often system-specific and time consuming to refine. Lee et al. (2014) overcome this issue by incorporating temporal variations in player performance, thus allowing for modeling change in a player's skill over time.

Pedersen et al. (2010) took a different tack, developing a neural network which charts three elements—behavioral characteristics of players, their self-reported emotions, and level parameters—in order to be able to successfully predict from in-game behavior and content what sort of affective experiences (enjoyment, frustration, effort, etc.) the players might report. This approach begins by studying play style, then constructs a model of users' preferences from which it produces levels to meet those desires. In so doing, they assume that a user's preferences remain fixed within a game session. However, if this assumption is suspect (or if a player's style of play might undergo variation), then a more adaptable model is needed to observe and describe how a player's behavior changes over time. Valls-Vargas et al. (2015) addressed this issue in proposing a new framework for player modeling, one which does not depend on the notion of fixed player style but instead attempts to predict a dynamic state of play. This framework captures fluctuations in player behavior through the use of episodic information and time interval models within a sequential machine learning method capable of learning several models progressively (such as a Support Vector Machine or Decision Tree). Testing various strategies of trace segmentation for predicting player style, they found that sequential machine learning approaches are superior to non-sequential ones when they include predictions from prior segments. Moreover, they found that predictive performance declines when the trace segmentation is either too detailed (minute-by-minute) or too imprecise (entire traces).

Factor analysis is also frequently employed in learning individual behavior. Zook and Riedl (2012) developed a tensor factorization technique for anticipating a player's performance in skill-based computer games. This data-driven tensor factorization method offers more precise adaptations of challenges to individual players because it is capable of forecasting alterations in a player's game mastery in time. They show via an empirical study (involving game users engaging a basic role-playing combat game) that tensor factorization models are both effective and easily scalable. But while their approach can even incorporate temporal behavior by including time factors, such approaches do not easily fit games which require predicting a fine granularity of action instead of predicting user's single valued performance.

After classification algorithms, various regression techniques coupled with clustering algorithms have been used in a pair of studies (see A1, A3, A4, A7, A12, A15, A27, A30, A31, A33, A35, A45, and A46 in Appendix A). These approaches primarily use clustering to obtain appropriate groupings of players and then perform a regression method within each group. Hence, we find the primary and secondary methods of player behavior modeling to be classification first, regression algorithms generally coupled with clustering techniques second.

Mahlmann et al. (2010), for example, considered whether one can forecast specific aspects of player behavior, examining by way of supervised learning the commercial game *Tomb Raider: Underworld* (TRU). Specifically, they sought to anticipate the moment at which a player will cease playing, or conversely how long the player would take to finish a game. The manufactured predictors focus on metrical data derived from game play in the two initial levels of TRU. The outcomes clearly demonstrate the efficacy of linear regression techniques as well as nonlinear approaches to classification.

Mere evolutionary algorithms were the least commonly applied data-driven approach for learning player behavior (see A2 and A39 in Appendix A). For instance, Bellotti et al. (2009) devised an engine for learning games derived from evolutionary computation approaches such as genetic algorithm and reinforcement learning (which separate expertise in game design from

domain expertise). Their method approaches the design of a learning game in terms of task authoring: domain experts identify and gloss domain knowledge to manifest within gameplay as tasks and task selection, while game designers establish the manner in which tasks are represented and chosen. A game experience module would then, at the start of a game, determine from a player's profile what subset of tasks to generate. In this manner, the game experience module takes on a double role: it becomes at once a domain expert (ascertaining a player's current capabilities and knowledge level) and a designer (selecting and staging suitable game tasks).

**3.3.2 Goal Recognition.** Abductive reasoning seeks to discern the user's intentions by studying his or her actions (Carberry 2001). This is referred to as goal recognition, and it views the user as enacting goal-directed behavior (or, broadly speaking, understands that he/she is aiming at realizing a specific state in the world). What goal recognition seeks to accomplish is to derive, from a user's prior actions and knowledge of the domain, an understanding of the state that the user seeks to realize in the world. Goal recognition is closely related to the challenges involved in other forms of recognition—specifically, of actions and plans. The former, also known as activity recognition (Turaga et al. 2008), deduces a user's action from sensory information (for instance, computer vision); whereas plan recognition (Kautz 1987) tackles the broader, more difficult challenge of predicting both a user's goal and the precise sequence of actions by which he or she will pursue it. In method, goal recognition can be categorized into alternate approaches—one rooted in planning systems, another rooted in models of probability. Each type of goal recognition possesses its own technological limitations, but that is a topic beyond the focus of this study.

Notably, player modeling significantly overlaps with goal/plan recognition, as such recognition entails a problem in which action sequences are interpreted in terms of the goal they are most probably attempting to realize. Plan recognition algorithms capture a set of observations and from it generates a set of goals that can account for the observed content. The smallest set of goals that can account for a sequence of actions is generally deemed superior to larger sets, since it probably holds more significant explanatory value (Carberry 2001). These methods of plan recognition are also applicable to forecasting player behavior: if a player's goal can be ascertained, it follows that the player will execute the intervening steps necessary for completing those goals. Player modeling differs from plan recognition, however, in that it sidesteps the process of predicting goals and deducing behavior from them and instead predicts behavior immediately.

**Table 4** demonstrates the importance of goal recognition in player modeling, as goal recognition constitutes the research objective of 12 studies. In these articles, the authors pursue the prediction or recognition of goals, plans, and actions (see A6, A8-A11, A13, A24-A26, A30, A32, and A41 in Appendix A). Several AI techniques have been studied and used for predicting goals, plans, and actions by the authors of reviewed research studies (see **Table 5**), such as classification, clustering, regression, and evolutionary algorithms. For each approach, as with behavior modeling, we have determined the primary or secondary AI techniques employed. In reviewing the studies given in **Table 5**, we discovered that a number of classification algorithms are used in goal/plan recognition. These include HMM, MLN, SVM, BN, LSTM, RNN, and DL. The classification algorithms employed most often for goal recognition were the MLN, HMM, and NN and its variant, which thus garnered increased attention concerning their modeling potential (see A6, A9, A13, A25, A26, and A41 in Appendix A). After these, the next in popularity were SVM (A10), BN (A32), and DL (A24). After classification algorithms, regression techniques along with clustering algorithms have also been used in a pair of studies (see A8 and A30 in Appendix A). Evolutionary algorithms, however, were not applied for goal recognition.

Ha et al. (2011), for instance, apply MLNs to develop a successful goal recognition framework for players. This framework employs model parameters derived from a corpus of player interactions

within a nonlinear game and enables an automatic goal recognition system superior to a number of baseline models. MLNs are also put to work in a similar goal recognition approach, by Baikadi et al. (2011), wherein problem-solving goals are linked to discovery events by means of probabilistic inference and first-order logical reasoning. When empirically tested, models employing discovery events-based models surpassed previous best approaches on all fronts. However, in using this approach—a combination of hand-authored logic formulae and machine-learned weights—some labor-intensive feature engineering efforts need to be eliminated (by utilizing, for example, multi-level feature abstraction techniques).

Another method is put forward in a study by Gold (2010), in which an Input-Output Hidden Markov Model (IOHMM) is trained to identify a player's goal within an action-adventure game. The goals—the hidden states in the IOHMM—were Fight, Explore, and Return to Town, and the observation model learned through players being directed toward specific goals and counting actions. Initially, there was little difference in performance between models trained to specific first-time players and the model trained to the experimenter. Subsequent models trained to these same players' ensuing gameplay vastly improved, however, over both the first-time player- and the experimenter-trained models. In other words, it appears that game goal recognition systems work best with players who have developed a specific playing style. It should be noted that the proposed approach by Gold (2010) was compared to a hand-authored finite state machine, a common computational framework used in commercial games, where the results showcase the effectiveness of the proposed approach; computational approaches based on deep learning for goal recognition, however, outperform the proposed approach (Min et al. 2014).

Furthermore, Min et al. (2014) developed a goal recognition framework grounded in stacked denoising autoencoders (a form of deep learning), in which the goal recognition models are trained through a corpus of player interactions. These models featured superior performance—significantly outperforming the best of the MLN-based goal recognition frameworks—and, moreover, present a profound advantage in that they require no labor-intensive engineering of features.

Laviers and Sukthankar (2011) have developed a technique for the real-time adaptation of plan repair policies, by means of Upper Confidence Bounds applied to Trees (UCT). Their research shows how such policies can, in a game of American football, be paired with plan recognition to generate a fully autonomous offense that can counter surprising shifts in defensive strategy. The offense produced by this real-time UCT approach easily surpassed both the baseline game and domain-specific heuristics in offensive efficiency, measured in average yardage and turnover avoidance. They cannot model goal alterations, however—that is, instances wherein a player adopts a new goal and forsakes the previous one. Moreover, this approach is unable to speak towards the relative superiority of training individual players versus aggregating player behavior, or the question of whether the usefulness of models depend on players having in-game experience.

Alternate approaches involve employing LSTM to learn these frameworks—goal recognition, multiple-goal recognition, task recognition, and plan repair policies. Regarding the use of latter in adversarial games, it is advisable to pair it with plan repair, for the reason that such games frequently require players to undertake a plan prior to being able to gauge the intentions of the opponent. This is especially true of multi-agent domains, in which it is not computationally possible to undertake total replanning. Hence, Min et al. (2016b) have approached goal recognition as a sequence labeling task by means of LSTM. Such an approach to goal recognition outperforms previous techniques (e.g., Markov's logic network-based models and n-gram encoded feed forward neural networks pre-trained with stacked denoising autoencoders) in accuracy testing. In this respect, LSTM-based approaches apparently solve a major problem in game player modeling, at once offering superior accuracy in goal recognition and also eliminating the exhaustive engineering previously required for such modeling.

### 3.4 Research Question 2

**RQ2:** What are the challenges of using data-driven approaches in game player modeling?

We present an overview in this section of major challenges to developing data-driven methods in game player modeling. In scrutinizing the reviewed works, we came to understand that data-driven approaches, while promising exciting advancements in game player modeling, present researchers with various challenges (see A1-A6, A12-A15, A18-A22, A24, A25, A27-A30, A33, A34, A36, A38, A39, A44, and A45 in Appendix A). These include:

- *Insufficient or Inaccessible Data:* Regardless of their specific area of focus in game player modeling, researchers struggle first and foremost with a dearth of publicly available, robust data. An expansive multimodal body of data and descriptions of gameplay and players is particularly urgent. Such a body of data must exhaustively cover gameplay data for a few highly populated multiplayer games—their actions and locations, events and timestamps. Moreover, researchers must be able to draw on information concerning players' demographics, questionnaire responses, and formal interviews (Of course, such data need not extend to every player in the database). The database should also incorporate the few significant databases of gameplay available at the present.
- *Heterogeneous Data:* While some machine learning methods, such as logistic regression, and ad hoc graphical models do promise certain advantages in analyzing and mining particular games data, they demand too much expert adjustment to be more widely applicable. There is also the challenge of interpreting a scarcity of raw data for sophisticated player modeling. This difficulty necessitates a comprehensive data-driven method, one which is not reliant on costly domain knowledge engineering and which produces a coherent and understandable model that is accurate both in its account of the data and in its prediction of game outcomes.
- *Problem of Generalizability:* The problem of generalizability describes a gaming situation in which an agent adapts only to a limited set of states in a game's world and is unable to generalize beyond them. It causes poor performance as soon as the game states change, or the agent engages new states. The standard technique in applied machine learning for preempting the tendency towards generalizability—a set of validation examples—might prove insufficient in a game environment, on account of both the time required to learn a set of validation examples and the difficulty of extracting such a set from a scarcity of training data.
- *Algorithmic Efficiency:* In highly populated and open-ended games, the massive amount of data collected can rapidly overwhelm players' PCs or gaming consoles. The solution to this technical difficulty requires a separation of modeling and predicting into two non-coterminous phases. A model can be developed offline, for instance, unconstrained by any untenable efficiency requirements, leaving only the prediction phase to be executed online with the efficiency necessary to prevent gameplay delays.
- *Data Scarcity Problem:* For a number of reasons, it is difficult to develop a set of training examples for games with a large search space, and particularly for games where a relevant example might be bound up with a series of delayed, interleaving, or compounding results. In such cases, player-generated material might have to provide the training examples, unless concrete targets can be removed from the learning process altogether. This scarcity of data can be offset, though, by joining methods of collaborative filtering with external information offered by content-based approaches.
- *Knowledge Engineering:* Multiple data-driven methods (e.g., Knowledge tracing/Bayesian network models) are unable to connect modeled player types to game events on their own, and must be supplemented by knowledge engineering annotations. Hence, these

approaches remain fundamentally reliant on expert authoring, in that expert authoring is necessary for task definition and the system-specific network models that generate the network structures that provide knowledge of users. This state of affairs is not sustainable. Data-driven methods must be constructed so as to cut down the amount of hand designing that goes into cognitive modeling and system-specific features.

– *Temporal Forecasting in Player Models*: Temporal forecasting is a central component of player predictions, since a player’s abilities and choices will fluctuate in time, and since such forecasting enables the development of sequential content. For this reason, even though most data-driven methods (e.g., collaborative filtering algorithms) are generally accurate, efficient, sufficient, and predictive, regardless of domain—especially in instances in which scarce data on individual players are offset by data gathered on other players—they must nevertheless factor for temporal performance variations so as to model a player’s changing abilities in time.

## 4 DISCUSSIONS

### 4.1 Research Question 3

**RQ3:** Looking forward, what are promising future directions in data-driven game player modeling?

Here we outline four promising future directions in the application of data-driven approaches in game player modeling (see A1, A5, A14, A19-A21, A25, A27-A30, A38, and A39 in Appendix A).

– *Data Mining Techniques for Individual Prediction*: While data-mining techniques have been shown to offer, via careful application, insight regarding the behaviors of groups, their utility in predicting individual behavior is still limited. Hence, player models generally offer only broad and fuzzy indications for ways in which a game should tailor itself to specific players (Yannakakis et al. 2013). One potential solution to this problem is to define a few player models and then categorizes individual players accordingly. In subsequent gameplay, the model can undergo small changes to better match the player. In other words, rather than viewing the model as a fixed representation of the player, by which the game makes adaptation decisions, it is seen as a dynamic representation of a group of players that modulates to match a specific player’s characteristics and thus initiates dynamic adaptation in the game. We believe that this second approach to game adaptation via player modeling offers practical advantages: it can quickly produce effective results in developing educational games that offer a personalized form of engagement to the player.

– *Hybrid Player Modeling Approaches*: In analyzing model-free and model-based methods of player modeling, we observe that model-free approaches manifest none of the argumentation and interpretation concerning the model’s choices that model-based approaches inevitably involve. And yet, model-based methods frequently overlook relevant features because their builders did not have sufficient insight, whereas model-free methods will automatically detect such features (Yannakakis 2012). The error of model-free approaches, however, is a tendency to deduce connections between user attributes, experience, and context that are entirely meaningless. These points to a broader present issue particularly in open-ended games: such games present the chance to collect and measure a set of player behavior features far more extensive than our current understanding of what these features might signify. Given this situation, model-free methods seem preferable, even though meaningful player models require domain-specific knowledge and the extraction and selection of features. We believe that one must characterize model-based and model-free approaches

as points at either end of a continuum, along which one can place various hybrid efforts to understand player behavior in games.

- Data-Driven Approaches to Conceptualizing Log Data:* While log files generated by games allow the researcher to learn behavior of players as they play the game, there are a number of practical issues associated with analyzing such data. To name a few, log files represent prohibitively large quantities of data; it is hard to interpret them since the responses of individual player are highly context dependent. It is also difficult to determine which actions represent key features of player performance given that log files are generally designed to capture all player actions relevant to game play, and not until after analysis can one know which of those actions were relevant to learning. For this reason, we hold that a data-driven approach offers significant promise to researchers, insofar as it does not depend on expensive domain knowledge engineering, and insofar as it can select and extract the essential conceptual features of player performance from games' log data.
- Data-Driven Approaches to Modeling Players in Infinitely Open and Replayable Worlds:* Interest has never been higher in the promise procedural content generation methods hold for optimized game design, in both commercial and independent game development. It is now assumed that new games will have more user-generated or procedurally-generated content than manual content, particularly because costly and bottlenecked content creation are major impediments in the game development process. But as automatically generated games become more common, it becomes increasingly difficult to model players in endlessly open worlds of infinite replayability value. Here also we are of the mind that researchers would significantly benefit from a data-driven approach that is not dependent on expensive domain knowledge engineering and can model players within such infinitely open and replayable worlds.

## 4.2 Limitations

We have constrained our references with the criteria of inclusion. Additionally, we have included, in applying assessment criteria, studies that addressed the work that initially generated the specific lines of research discussed. Readers must be advised that it is impossible to exhaustively review, in a single article, the myriad aspects of data-driven methods in game player modeling. In fact, certain topics (such as procedural content generation) are deserving of an entire review on their own. Rather, our aim here has been to give a representative selection of current research within each learning method.

## 4.3 Conclusions

In spite of the increased interest in data-driven approaches to game player modeling, there has yet to be any effort to review current empirical evidence regarding the benefits, challenges, and applications of such approaches. This article has thus focused on conclusions from empirical studies, thereby offering a systematic overview of the evidence pertaining to data-driven approaches in game player modeling. By examining the available literature for representative, rigorous, and frequently cited case studies from game player modeling domains, we cast light on the data-driven approaches that have been employed and in so doing demonstrate the potential held by this new field in game research. In Table 6, we present our findings concerning the aims of data-driven approaches to player modeling in games, the algorithms employed in this type of player modeling, and the present difficulties and future directions of game player modeling.

While previous methods relied on questionnaires to gain insight into perceptions and attitudes, now every movement and “click” in an electronic learning environment might encode useful information that can be tracked and interpreted. Computational methods offer the potential to isolate,

**Table 6. Objectives, Techniques, Current Challenges, and Future Directions of Data-Driven Approaches to Game Player Modeling**

| Research Objectives of the Reviewed Empirical Evidence                              |  |
|---|--|
| Player behavior modeling (experience modeling/procedural content generation)        |  |
| Goal/plan recognition   |  |
| Data-driven Techniques in Game Player Modeling                                      |  |
| Classification  |  |
| Clustering  |  |
| Regression  |  |
| Evolutionary Algorithms   |  |
| Current Challenges  |  |
| Insufficient or inaccessible data   |  |
| Heterogeneous data  |  |
| Problem of generalizability   |  |
| Algorithmic efficiency  |  |
| Data scarcity problem   |  |
| Knowledge engineering   |  |
| Temporal forecasting in player models   |  |
| Future Directions   |  |
| Data mining techniques for individual prediction                                    |  |
| Hybrid player modeling approaches   |  |
| Data-driven approaches to conceptualizing log data                                  |  |
| Data-driven approaches to modeling players in infinitely open and replayable worlds |  |

identify, and categorize any simple or complex action within a game, placing it within meaningful patterns. Interactions of all sorts can be coded into behavioral patterns and interpreted to guide decision-making. This exciting juncture lies at the crossroads of computer and learning science, psychology, and pedagogy. What remains is to grasp the deeper learning processes by means of breaking them down into simpler and discrete mechanisms. It is our hope and belief that this active area of research will continue to provide invaluable contributions to the development of dynamic, powerful, and accurate games, both for designers and for players.

## APPENDIX A

Demographic data and overview of the selected studies:

## ACKNOWLEDGMENTS

Special thanks to the anonymous reviewers for their insightful comments that helped us make this article better.

## REFERENCES

- Kostas Anagnostou and Manolis Maragoudakis. 2009. Data mining for player modeling in videogames. In *Proceedings of the 13th Panhellenic Conference on Informatics (PCI'09)*. IEEE, 30–34.
- Alok Baikadi, Jonathan Rowe, Bradford Mott, and James Lester. 2011. Generalizability of goal recognition models in narrative-centered learning environments. In *Proceedings of the International Conference on User Modeling, Adaptation, and Personalization*. Springer International Publishing, 278–289.
- Sander Bakkes, Pieter H. M. Spronck, and Giel van Lankveld. 2012. Player behavioural modeling for video games. *Entertainment Computing*, 3, 3, 71–79. DOI: <http://dx.doi.org/10.1016/j.entcom.2011.12.001>

- Francesco Bellotti, Riccardo Berta, Alessandro De Gloria, and Ludovica Primavera. 2009. Adaptive experience engine for serious games. *IEEE Transactions on Computational Intelligence and AI in Games* 1, 4, 264–280.
- Francis Bisson, Hugo Larochelle, and Frouduald Kabanza. 2015. Using a recursive neural network to learn an agent's decision model for plan recognition. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI)*. 918–924.
- Zoheb Borbora and Jaideep Srivastava. 2012. User behavior modeling approach for churn prediction in online games. In *Proceedings of the 2012 International Conference on Privacy, Security, Risk and Trust (PASSAT) and Proceedings of the 2012 International Conference on Social Computing (SocialCom)*. IEEE, 51–60.
- Eric Butler, Erik Andersen, Adam M. Smith, Sumit Gulwani, and Zoran Popović. 2015. Automatic game progression design through analysis of solution features. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, 2407–2416.
- Sandra Carberry. 2001. Techniques for plan recognition. *User Modeling and User-Adapted Interaction* 11, 1, 31–48.
- Ben Cowley, Darryl Charles, Michaela Black, and Ray Hickey. 2009. Analyzing player behavior in Pacman using feature-driven decision theoretic predictive modeling. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games (CIG 2009)*. IEEE, 170–177.
- Benjamin Cowley, Darryl Charles, Michaela Black, and Ray Hickey. 2013. Real-time rule-based classification of player types in computer games. *User Modeling and User-Adapted Interaction* 23, 5, 489–526.
- Benjamin Cowley, Marco Filetti, Kristian Lukander, Jari Torniainen, Andreas Henelius, Lauri Ahonen, Oswald Barral 2016. The psychophysiology primer: A guide to methods and a broad review with a focus on human-computer interaction. *Foundations and Trends® Human–Computer Interaction* 9, 3–4, 151–308.
- Benjamin Cowley, Ilkka Kosunen, Petri Lankoski, J. Matias Kivikangas, Simo Järvelä, Inger Ekman, Jaakko Kemppainen, and Niklas Ravaja. 2014. Experience assessment and design in the analysis of gameplay. *Simulation & Gaming* 45, 1, 41–69.
- Anders Drachen, Alessandro Canossa, and Georgios N. Yannakakis. 2009. Player modeling using self-organization in Tomb Raider: Underworld. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games (CIG 2009)*. IEEE, 1–8.
- Marlon Etheredge, Ricardo Lopes, and Rafael Bidarra. 2013. A generic method for classification of player behavior. In *Proceedings of the 2nd Workshop on Artificial Intelligence in the Game Design Process*. 1–7.
- Mohammad Falakmasir, José P. González-Brenes, Geoffrey J. Gordon, and Kristen E. DiCerbo. 2016. A data-driven approach for inferring student proficiency from game activity logs. In *Proceedings of the 3rd ACM Conference on Learning@ Scale*. ACM, 341–349.
- Leo Galway, Darryl Charles, and Michaela Black. 2008. Machine learning in digital games: A survey. *Artificial Intelligence Review* 29, 2, 123–161.
- Chen Gao, Haifeng Shen, and M. Ali Babar. 2016. Concealing jitter in multi-player online games through predictive behaviour modeling. In *Proceedings of the IEEE 20th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*. IEEE, 62–67.
- Kevin Gold. 2010. Training goal recognition online from low-level inputs in an action-adventure game. In *Proceedings of AIIDE*.
- Jeremy Gow, Robin Baumgarten, Paul Cairns, Simon Colton, and Paul Miller. 2012. Unsupervised modeling of player style with LDA. *IEEE Transactions on Computational Intelligence and AI in Games* 4, 3, 152–166.
- Corrado Grappiolo, Yun-Gyung Cheong, Julian Togelius, Rilla Khaled, and Georgios N. Yannakakis. 2011. Towards player adaptivity in a serious game for conflict resolution. In *Proceedings of the 3rd International Conference on Games and Virtual Worlds for Serious Applications (VS-GAMES)*. IEEE, 192–198.
- Eunyoung Ha, Jonathan P. Rowe, Bradford W. Mott, and James C. Lester. 2011. Goal recognition with Markov logic networks for player-adaptive games. In *Proceedings of AIIDE*.
- Erik Harpstead, Thomas Zimmermann, Nachiappan Nagapan, Jose J. Guajardo, Ryan Cooper, Tyson Solberg, and Dan Greenawalt. 2015. What drives people: Creating engagement profiles of players from game log data. In *Proceedings of the 2015 Annual Symposium on Computer-Human Interaction in Play*. ACM, 369–379.
- Brent Harrison and David L. Roberts. 2011. Using sequential observations to model and predict player behavior. In *Proceedings of the 6th International Conference on Foundations of Digital Games*. ACM, 91–98.
- Frouduald Kabanza, Philippe Bellefeuille, Francis Bisson, Abder Rezak Benaskeur, and Hengameh Irandoust. 2010. Opponent behaviour recognition for real-time strategy games. *Plan, Activity, and Intent Recognition* 10, 5.
- Henry Kautz. 1987. *A Formal Theory of Plan Recognition*. Ph.D. Dissertation, Bell Laboratories.
- Keele Staffs. 2007. *Guidelines for Performing Systematic Literature Reviews in Software Engineering*. EBSE Technical Report, Ver. 2.3. EBSE. sn.
- Kennard Lavers and Gita Sukthankar. 2011. A real-time opponent modeling system for rush football. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 22, 3, 2476.

- Jae Lee Seong, Yun-En Liu, and Zoran Popovic. 2014. Learning individual behavior in an educational game: a data-driven approach. In *Proceedings of the Conference on Educational Data Mining 2014*. 114–121.
- Yun-En Liu, Travis Mandel, Eric Butler, Erik Andersen, Eleanor O'Rourke, Emma Brunsell, and Zoran Popovic. 2013. Predicting player moves in an educational game: A hybrid approach. In *Proceedings of the Conference on Educational Data Mining 2013*.
- Simon Lucas, Michael Mateas, Mike Preuss, Pieter Spronck, Julian Togelius, Peter I. Cowling, Michael Buro, Michal Bida, Adi Botea, and Bruno Bouzy. 2013. Artificial and computational intelligence in games. *Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik*.
- Linbo Luo, Haiyan Yin, Wentong Cai, Jinghui Zhong, and Michael Lees. 2016. Design and evaluation of a data-driven scenario generation framework for game-based training. *IEEE Transactions on Computational Intelligence and AI in Games* 99, 1–1.
- Marlos Machado, Eduardo P. C. Fantini, and Luiz Chaimowicz. 2011. Player modeling: Towards a common taxonomy. In *Proceedings of the 16th IEEE International Conference on Computer Games (CGAMES)*. IEEE, 50–57.
- Tobias Mahlmann, Anders Drachen, Julian Togelius, Alessandro Canossa, and Georgios N. Yannakakis. 2010. Predicting player behavior in Tomb Raider: Underworld. In *Proceedings of the 2010 IEEE Symposium on Computational Intelligence and Games (CIG)*. IEEE, 178–185.
- Wookhee Min, Alok Baikadi, Bradford Mott, Jonathan Rowe, Barry Liu, Eun Young Ha, and James Lester. 2016a. A generalized multidimensional evaluation framework for player goal recognition. In *Proceedings of the 12th Annual AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*.
- Bradford Mott, Jonathan Rowe, Barry Liu, and James Lester. 2016b. Player goal recognition in open-world digital games with long short-term memory networks. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*. 2590–2596.
- Wookhee Min, Eunyoung Ha, Jonathan P. Rowe, Bradford W. Mott, and James C. Lester. 2014. Deep learning-based goal recognition in open-ended digital games. In *Proceedings of AIIDE*.
- and Thomas Gärtner. 2009. Player modeling for intelligent difficulty adjustment. In *Proceedings of the International Conference on Discovery Science*. Springer, Berlin, 197–211.
- Jeff Orkin, Tynan Smith, Hilke Reckman, and Deb Roy. 2010. Semi-automatic task recognition for interactive narratives with EAT & RUN. In *Proceedings of the Intelligent Narrative Technologies Iii Workshop*. ACM.
- Christopher Pedersen, Julian Togelius, and Georgios N. Yannakakis. 2010. Modeling player experience for content creation. *IEEE Transactions on Computational Intelligence and AI in Games* 2, 1, 54–67.
- Daniel Ramirez-Cano, Simon Colton, and Robin Baumgarten. 2010. Player classification using a meta-clustering approach. In *Proceedings of the 3rd Annual International Conference Computer Games, Multimedia & Allied Technology*. 297–304.
- Noor Shaker, Georgios N. Yannakakis, and Julian Togelius. 2010. Towards automatic personalized content generation for platform games. In *Proceedings of AIIDE*.
- Manu Sharma, Santiago Ontañón, Manish Mehta, and Ashwin Ram. 2010. Drama management and player modeling for interactive fiction games. *Computational Intelligence* 26, 2, 183–211.
- Pieter Spronck and Freek den Teuling. 2010. Player modeling in civilization IV. In *Proceedings of AIIDE*.
- Gabriel Synnaeve and Pierre Bessiere. 2011. A Bayesian model for plan recognition in RTS games applied to StarCraft. arXiv preprint arXiv:1111.3735.
- Marco Tamassia, William Raffe, Rafet Sifa, Anders Drachen, Fabio Zambetta, and Michael Hitchens. 2016. Predicting player churn in destiny: A hidden Markov models approach to predicting player departure in a major online game. In *Proceedings of the IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 1–8.
- Pavan Turaga, Rama Chellappa, Venkatramana S. Subrahmanian, and Octavian Udrea. 2008. Machine recognition of human activities: A survey. *IEEE Transactions on Circuits and Systems for Video Technology* 18, 11, 1473–1488.
- Josep Valls-Vargas, Santiago Ontañón, and Jichen Zhu. 2015. Exploring player trace segmentation for dynamic play style prediction. In *Proceedings of the 11th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*. 93–99.
- Ben George Weber and Michael Mateas. 2009. A data mining approach to strategy prediction. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games (CIG 2009)*. IEEE, 140–147.
- Ben George Weber, Michael Mateas, and Arnav Jhala. 2011a. Using data mining to model player experience. In *Proceedings of the FDG Workshop on Evaluating Player Experience in Games*.
- Ben George Weber, Michael John, Michael Mateas, and Arnav Jhala. 2011b. Modeling player retention in Madden NFL 11. In *Proceedings of IAAI*. 1701–1706.
- Hanting Xie, Daniel Kudenko, Sam Devlin, and Peter Cowling. 2014. Predicting player disengagement in online games. In *Proceedings of the Workshop on Computer Games*. Springer International Publishing, 133–149.
- Georgios N. Yannakakis and John Hallam. 2009. Real-time game adaptation for optimizing player satisfaction. *IEEE Transactions on Computational Intelligence and AI in Games* 1, 2, 121–133.

- Georgios N. Yannakakis, Pieter Spronck, Daniele Loiacono, and Elisabeth André. 2013. Player modeling. In *Dagstuhl Follow-Ups*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.
- Georgios N. Yannakakis 2012. Game AI revisited. In *Proceedings of the 9th Conference on Computing Frontiers*. ACM, 285-292.
- Hong Yu and Mark O. Riedl. 2012. A sequential recommendation approach for interactive personalized story generation. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 71-78.
- Alexander Zook, Stephen Lee-Urban, Michael R. Drinkwater, and Mark O. Riedl. 2012. Skill-based mission generation: A data-driven temporal player modeling approach. In *Proceedings of the 3rd Workshop on Procedural Content Generation in Games*. ACM, 6.
- Alexander Zook and Mark O. Riedl. 2012. A temporal data-driven player model for dynamic difficulty adjustment. In *Proceedings of AIIDE*.

Received October 2016; revised May 2017; accepted September 2017

Georgios N. Yannakakis and Julian Togelius

# Artificial Intelligence and Games

January 26, 2018

Springer



*To our families*



## Foreword

It is my great pleasure to write the foreword for this excellent and timely book. Games have long been seen as the perfect test-bed for artificial intelligence (AI) methods, and are also becoming an increasingly important application area. Game AI is a broad field, covering everything from the challenge of making super-human AI for difficult games such as Go or *StarCraft*, to creative applications such as the automated generation of novel games.

Game AI is as old as AI itself, but over the last decade the field has seen massive expansion and enrichment with the inclusion of video games, which now comprise more than 50% of all published work in the area and enable us to address a broader range of challenges that have great commercial, social, economic and scientific interest. A great surge in research output occurred in 2005, coinciding with both the first IEEE Symposium (Conference) on Computational Intelligence and Games (CIG)—which I co-chaired with Graham Kendall—and the first AAAI AIIDE Conference (Artificial Intelligence in Digital Entertainment). Since then this rich area of research has been more explored and better understood. The Game AI community pioneered much of the research which is now becoming (or about to become) more mainstream AI, such as Monte Carlo Tree Search, procedural content generation, playing games based on screen capture, and automated game design.

Over the last decade, progress in deep learning has had a profound and transformational effect on many difficult problems, including speech recognition, machine translation, natural language understanding and computer vision. As a result, computers can now achieve human-competitive performance in a wide range of perception and recognition tasks. Many of these systems are now available to the programmer via a range of so-called cognitive services. More recently, deep reinforcement learning has achieved ground-breaking success in a number of difficult challenges, including Go and the amazing feat of learning to play games directly from screen capture (playing from pixels). It is fascinating to contemplate what this could mean for games as we stumble towards human-level intelligence in an increasing number of areas. The impacts will be significant for the intelligence of in-game characters, the way in which we interact with them and for the way games are designed and tested.

This book makes an enormous contribution to this captivating, vibrant area of study: an area that is developing rapidly both in breadth and depth as AI is able to cope with a wider range of tasks (and to perform those tasks to increasing levels of excellence). The service to the community will be felt for many years to come: the book provides an easier and more comprehensive entry point for newcomers to the field than previously available, whilst also providing an indispensable reference for existing AI and Games researchers wishing to learn about topics outside their direct field of interest.

Georgios Yannakakis and Julian Togelius have been involved with the field ever since its widespread expansion to video games, and they both presented papers at the first 2005 CIG. Over the years they have made an enormous contribution to the field with a great number of highly cited papers presenting both novel research and comprehensive surveys. It is my opinion that these authors are best qualified to write this book, and they do not disappoint. The book will serve the community very well for many years to come.

London, August 2017

*Simon Lucas*

# Preface

*Of all the things that wisdom provides for the complete happiness of one's entire life, by far the greatest is friendship.*

Epicurus, *Principal Doctrines*, 27

*Human beings, viewed as behaving systems, are quite simple. The apparent complexity of our behavior over time is largely a reflection of the complexity of the environment in which we find ourselves.*

Herbert A. Simon

It would be an understatement to say that **Artificial Intelligence** (AI) is a popular topic at the moment, and it is unlikely to become any less important in the future. More researchers than ever work on AI in some form, and more non-researchers than ever are interested in the field. It would also be an understatement to say that **games** are a popular application area for AI research. While board games have been central to AI research since the inception of the field, video games have during the last decade increasingly become the domain of choice for testing and showcasing new algorithms. At the same time, video games themselves have become more diverse and sophisticated, and some of them incorporate advances in AI for controlling non-player characters, generating content or adapting to players. Game developers have increasingly realized the power of AI methods to analyze large volumes of player data and optimize game designs. And a small but growing community of researchers and designers experiment with ways of using AI to design and create complete games, automatically or in dialog with humans. It is indeed an exciting time to be working on AI and games!

This is a book about **AI and games**. As far as we know, it is the first *comprehensive* textbook covering the field. With comprehensive, we mean that it features all the major application areas of AI methods within games: game-playing, content generation and player modeling. We also mean that it discusses AI problems in many different types of games, including board games and video games of many genres. The book is also comprehensive in that it takes multiple perspectives of AI and games: how games can be used to test and develop AI, how AI can be used

to make games better and easier to develop, and to understand players and design. While this is an academic book which is primarily aimed at students and researchers, we will frequently address problems and methods relevant for game designers and developers.

We wrote this book based on our long experience doing research on AI for games, each on our own and together, and helping lead and shape the research community. We both independently started researching AI methods in games in 2004, and we have been working together since 2009. Together, we played a role in introducing research topics such as procedural content generation and player modeling to the academic research community, and created several of the most widely used game-based AI benchmarks. This book is in a sense a natural outgrowth of the classes on AI and games we have taught at three universities, and the several survey papers of the field and of individual research topics within it that we have published over the years. But the book is also a response to the lack of a good introductory book for the research field. Early discussions on writing such a book date back at least a decade, but no-one actually wrote one, until now.

It could be useful to point out what this book is not. It is not a hands-on book with step-by-step instructions on how to build AI for your game. It does not feature discussions of any particular game engine or software framework, and it does not discuss software engineering aspects or many implementation aspects at all. It is not an introductory book, and it does not give a gentle introduction to basic AI or game design concepts. For all these roles, there are better books available.

Instead, this is a book for readers who already understand AI methods and concepts to the level of having taken an introductory AI course, and the introductory computer science or engineering courses that led up to that course. The book assumes that the reader is comfortable with reading a pseudocode description of an algorithm and implementing it. Chapter 2 is a summary of AI methods used in the book, but is intended more as a reference and refresher than as an introduction. The book also assumes a basic familiarity with games, if not designing them then at least playing them.

The use case for this textbook that we had in mind when writing it is for a one- or a two-semester graduate-level or advanced undergraduate level class. This can take several different shapes to support different pedagogical practices. One way of teaching such a class would be a traditional class, with lectures covering the chapters of the book in order, a conventional pen-and-paper exam at the end, and a small handful of programming exercises. For your convenience, each of the main chapters of the book include suggestions for such exercises. Another way of organizing a class around this book, more in line with how we personally prefer to teach such courses, is to teach the course material during the first half of the semester and spend the second half on a group project.

The material offered by this book can be used in various ways and, thus, support a number of different classes. In our experience, a traditional two-semester class on game artificial intelligence would normally cover Chapter 2 and Chapter 3 in the first semester and then focus on alternative uses of AI in games (Chapters 4 and 5) in the second semester. When teaching the material in compressed (one-semester) fash-

ion instead, it is advisable to skip Chapter 2 (using it as a reference when needed), and focus the majority of the lectures on Chapters 3, 4 and 5. Chapters 6 and 7 can be used as material for inspiring advanced graduate-level projects in the area. Beyond the strict limits of game AI, Chapter 4 (or sections of it) can complement classes with a focus on game design or computational creativity whereas Chapter 5 can complement classes with a focus on affective computing, user experience, and data mining. It is of course also possible to use this book for an introductory undergraduate class for students who have not taken an AI class before, but in that case we advise the instructor to select a small subset of topics to focus on, and to complement the book with online tutorials on specific methods (e.g., best-first search, evolutionary computation) that introduce these topics in a more gentle fashion than this book does.

Chania, Crete, Greece  
New York, NY, USA

*Georgios N. Yannakakis  
Julian Togelius*

September 2017



## Acknowledgments

Writing a book of this size would be impossible without the support and the contributions of a large number of people. First, we would like to thank Ronan Nugent, our editor at Springer, who guided us and helped us from the book proposal phase all the way to its final production.

We also would like to thank the following persons who read all (or parts of a draft of) the book and provided useful feedback: Amy Hoover, Amin Babadi, Sander Bakkes, Vadim Bulitko, Phil Carlisle, Georgios Chalkiadakis, Dave Churchill, Mike Cook, Renato Cunha, Kevin Dill, Nathaniel Du Preez-Wilkinson, Chris Elion, Andy Elmsley, David Fogel, Bernardo Galvão, Kazu-ma Hashimoto, Aaron Isaksen, Emil Johansen, Mark Jones, Niels Justesen, Graham Kendall, Jakub Kowalski, Antonios Liapis, Nir Lipovetzky, Jhovan Mauricio López, Simon Lucas, Jacek Mańdziuk, Luciana Mariñelarena-Dondena, Chris Martens, Sean Mealin, Mark Nelson, Sven Neuhaus, Alexander Osherenko, Santiago Ontañón, Cale Plut, Mike Preuss, Hartmut Procha-ska, Christoffer Holmgård, Florian Richoux, Sebastian Risi, Christoph Salge, Andrea Schiel, Jacob Schrum, Magy Seif El-Nasr, Adam Smith, Gillian Smith, Dennis Soemers, Nathan Sturtevant, Gabriel Synnaeve, Nicolas Szilas, Varunyu Vorachart, James Wen, Marco Wiering, Mark Winands, Junkai Lu, Francesco Calimeri, Diego Pérez Liébana, Corine Jacobs, Junkai Lu, Hana Rudova, and Robert Zubek. Of these, we wish to especially thank Simon Lucas for writing the foreword of the book, Georgios Chalkiadakis for providing substantial input on parts of the book related to game theory, and Mark Nelson, Antonios Liapis, Mike Preuss and Adam Smith for reviewing large parts of the book and providing particularly detailed feedback. We also want to thank all those who granted us their permission to reproduce images and figures from their papers; they are all acknowledged in the figure captions of the book. Special thanks also go to Rebecca Portelli and Daniel Mercieca for the artwork featured on the cover page of the book.

Some chapters of this book build on papers or other book chapters that we have co-authored. In some cases the papers are co-authored by more than the two of us; for those papers our co-authors graciously gave us permission to reuse parts of the material and we wish to thank them for that. In particular,

- Chapter 1: [764, 700].
- Chapter 4: Chapter 2 and Chapter 3 from [616], and [381].
- Chapter 5: [778, 176, 782, 781].
- Chapter 6: [785].
- Chapter 7: [718, 458].

Writing this book has been a very long journey for both of us; a challenging and overwhelming journey at times. There are a number of people who have supported this effort that we would like to thank. Georgios wishes to thank the people at NYU Tandon School of Engineering for hosting him during the early days of book planning and the people at the Technical University of Crete for hosting him during the later stages of this project. Georgios would also like to thank the University of Malta for granting him a sabbatical leave, without which the book would not have been possible. Georgios and Julian also wish to thank each other for putting up with each other despite everything, and announce that they intend to buy each other drinks to celebrate when they meet next.

Last but not least, both of us wish to thank our families and all the people who showed us their support, care, encouragement and love at times when they were absolutely needed. You are too many to list here, but big thanks go to you all! Georgios would not have been able to write this book without the love and support of his family: Amyrsa and Myrto have been his core inspiration, Stavroula has been the main driving force behind the writing at all times; this book is dedicated to you!

# Contents

## Part I Background

|          |  |    |
|----------|--|----|
| <b>1</b> | <b>Introduction</b>                                  | 3  |
| 1.1      | This Book  | 4  |
| 1.1.1    | Why Did We Write This Book?                          | 5  |
| 1.1.2    | Who Should Read This Book?                           | 6  |
| 1.1.3    | A Short Note on Terminology                          | 7  |
| 1.2      | A Brief History of Artificial Intelligence and Games | 8  |
| 1.2.1    | Academia   | 10 |
| 1.2.2    | Industry   | 11 |
| 1.2.3    | The “Gap”  | 13 |
| 1.3      | Why Games for Artificial Intelligence                | 15 |
| 1.3.1    | Games Are Hard and Interesting Problems              | 15 |
| 1.3.2    | Rich Human-Computer Interaction                      | 17 |
| 1.3.3    | Games Are Popular                                    | 18 |
| 1.3.4    | There Are Challenges for All AI Areas                | 20 |
| 1.3.5    | Games Best Realize Long-Term Goals of AI             | 21 |
| 1.4      | Why Artificial Intelligence for Games                | 23 |
| 1.4.1    | AI Plays and Improves Your Game                      | 23 |
| 1.4.2    | More Content, Better Content                         | 24 |
| 1.4.3    | Player Experience and Behavioral Data Analytics      | 25 |
| 1.5      | Structure of This Book                               | 25 |
| 1.5.1    | What We (Don’t) Cover in This Book                   | 26 |
| 1.6      | Summary  | 28 |
| <b>2</b> | <b>AI Methods</b>                                    | 29 |
| 2.1      | General Notes  | 30 |
| 2.1.1    | Representation                                       | 30 |
| 2.1.2    | Utility  | 31 |
| 2.1.3    | Learning = Maximize Utility (Representation)         | 32 |
| 2.2      | Ad-Hoc Behavior Authoring                            | 32 |

---

|       |   |    |
|-------|---|----|
| 2.2.1 | Finite State Machines . . . . .                       | 33 |
| 2.2.2 | Behavior Trees . . . . .                              | 34 |
| 2.2.3 | Utility-Based AI . . . . .                            | 37 |
| 2.2.4 | Further Reading . . . . .                             | 39 |
| 2.3   | Tree Search . . . . .                                 | 39 |
| 2.3.1 | Uninformed Search . . . . .                           | 40 |
| 2.3.2 | Best-First Search . . . . .                           | 41 |
| 2.3.3 | Minimax . . . . .                                     | 42 |
| 2.3.4 | Monte Carlo Tree Search . . . . .                     | 45 |
| 2.3.5 | Further Reading . . . . .                             | 49 |
| 2.4   | Evolutionary Computation . . . . .                    | 49 |
| 2.4.1 | Local Search . . . . .                                | 50 |
| 2.4.2 | Evolutionary Algorithms . . . . .                     | 52 |
| 2.4.3 | Further Reading . . . . .                             | 57 |
| 2.5   | Supervised Learning . . . . .                         | 57 |
| 2.5.1 | Artificial Neural Networks . . . . .                  | 59 |
| 2.5.2 | Support Vector Machines . . . . .                     | 66 |
| 2.5.3 | Decision Tree Learning . . . . .                      | 68 |
| 2.5.4 | Further Reading . . . . .                             | 71 |
| 2.6   | Reinforcement Learning . . . . .                      | 71 |
| 2.6.1 | Core Concepts and a High-Level Taxonomy . . . . .     | 73 |
| 2.6.2 | Q-Learning . . . . .                                  | 74 |
| 2.6.3 | Further Reading . . . . .                             | 76 |
| 2.7   | Unsupervised Learning . . . . .                       | 77 |
| 2.7.1 | Clustering . . . . .                                  | 77 |
| 2.7.2 | Frequent Pattern Mining . . . . .                     | 80 |
| 2.7.3 | Further Reading . . . . .                             | 82 |
| 2.8   | Notable Hybrid Algorithms . . . . .                   | 82 |
| 2.8.1 | Neuroevolution . . . . .                              | 83 |
| 2.8.2 | TD Learning with ANN Function Approximators . . . . . | 84 |
| 2.8.3 | Further Reading . . . . .                             | 87 |
| 2.9   | Summary . . . . .                                     | 88 |

## **Part II Ways of Using AI in Games**

|       |   |     |
|-------|---|-----|
| 3     | Playing Games . . . . .                               | 91  |
| 3.1   | Why Use AI to Play Games? . . . . .                   | 92  |
| 3.1.1 | Playing to Win in the Player Role . . . . .           | 93  |
| 3.1.2 | Playing to Win in a Non-player Role . . . . .         | 95  |
| 3.1.3 | Playing for Experience in the Player Role . . . . .   | 95  |
| 3.1.4 | Playing for Experience in a Non-player Role . . . . . | 96  |
| 3.1.5 | Summary of AI Game-Playing Goals and Roles . . . . .  | 98  |
| 3.2   | Game Design and AI Design Considerations . . . . .    | 98  |
| 3.2.1 | Characteristics of Games . . . . .                    | 98  |
| 3.2.2 | Characteristics of AI Algorithm Design . . . . .      | 105 |

|          |   |            |
|----------|---|------------|
| 3.3      | How Can AI Play Games? . . . . .                | 109        |
| 3.3.1    | Planning-Based Approaches . . . . .             | 109        |
| 3.3.2    | Reinforcement Learning . . . . .                | 115        |
| 3.3.3    | Supervised Learning . . . . .                   | 118        |
| 3.3.4    | Chimeric Game Players . . . . .                 | 118        |
| 3.4      | Which Games Can AI Play? . . . . .              | 119        |
| 3.4.1    | Board Games . . . . .                           | 119        |
| 3.4.2    | Card Games . . . . .                            | 121        |
| 3.4.3    | Classic Arcade Games . . . . .                  | 123        |
| 3.4.4    | Strategy Games . . . . .                        | 132        |
| 3.4.5    | Racing Games . . . . .                          | 136        |
| 3.4.6    | Shooters and Other First-Person Games . . . . . | 139        |
| 3.4.7    | Serious Games . . . . .                         | 141        |
| 3.4.8    | Interactive Fiction . . . . .                   | 144        |
| 3.4.9    | Other Games . . . . .                           | 145        |
| 3.5      | Further Reading . . . . .                       | 148        |
| 3.6      | Exercises . . . . .                             | 149        |
| 3.6.1    | Why Ms Pac-Man? . . . . .                       | 149        |
| 3.7      | Summary . . . . .                               | 150        |
| <b>4</b> | <b>Generating Content . . . . .</b>             | <b>151</b> |
| 4.1      | Why Generate Content? . . . . .                 | 152        |
| 4.2      | Taxonomy . . . . .                              | 154        |
| 4.2.1    | Taxonomy for Content . . . . .                  | 154        |
| 4.2.2    | Taxonomy for Methods . . . . .                  | 155        |
| 4.2.3    | Taxonomy of Roles . . . . .                     | 156        |
| 4.3      | How Could We Generate Content? . . . . .        | 157        |
| 4.3.1    | Search-Based Methods . . . . .                  | 157        |
| 4.3.2    | Solver-Based Methods . . . . .                  | 161        |
| 4.3.3    | Grammar-Based Methods . . . . .                 | 162        |
| 4.3.4    | Cellular Automata . . . . .                     | 165        |
| 4.3.5    | Noise and Fractals . . . . .                    | 169        |
| 4.3.6    | Machine Learning . . . . .                      | 171        |
| 4.4      | Roles of PCG in Games . . . . .                 | 175        |
| 4.4.1    | Mixed-Initiative . . . . .                      | 176        |
| 4.4.2    | Autonomous . . . . .                            | 180        |
| 4.4.3    | Experience-Driven . . . . .                     | 180        |
| 4.4.4    | Experience-Agnostic . . . . .                   | 182        |
| 4.5      | What Could Be Generated? . . . . .              | 184        |
| 4.5.1    | Levels and Maps . . . . .                       | 184        |
| 4.5.2    | Visuals . . . . .                               | 186        |
| 4.5.3    | Audio . . . . .                                 | 187        |
| 4.5.4    | Narrative . . . . .                             | 189        |
| 4.5.5    | Rules and Mechanics . . . . .                   | 191        |
| 4.5.6    | Games . . . . .                                 | 193        |

---

|          |  |            |
|----------|--|------------|
| 4.6      | Evaluating Content Generators .....              | 197        |
| 4.6.1    | Why Is It Difficult? .....                       | 197        |
| 4.6.2    | Function vs. Aesthetics .....                    | 197        |
| 4.6.3    | How Can We Evaluate a Generator? .....           | 198        |
| 4.7      | Further Reading .....                            | 201        |
| 4.8      | Exercises .....                                  | 201        |
| 4.8.1    | Maze Generation .....                            | 201        |
| 4.8.2    | Platformer Level Generation .....                | 201        |
| 4.9      | Summary .....                                    | 202        |
| <b>5</b> | <b>Modeling Players .....</b>                    | <b>203</b> |
| 5.1      | What Player Modeling Is and What It Is Not ..... | 204        |
| 5.2      | Why Model Players? .....                         | 206        |
| 5.3      | A High-Level Taxonomy of Approaches .....        | 207        |
| 5.3.1    | Model-Based (Top-Down) Approaches .....          | 208        |
| 5.3.2    | Model-Free (Bottom-Up) Approaches .....          | 212        |
| 5.3.3    | Hybrids .....                                    | 213        |
| 5.4      | What Is the Model's Input Like? .....            | 213        |
| 5.4.1    | Gameplay .....                                   | 214        |
| 5.4.2    | Objective .....                                  | 214        |
| 5.4.3    | Game Context .....                               | 217        |
| 5.4.4    | Player Profile .....                             | 218        |
| 5.4.5    | Linked Data .....                                | 219        |
| 5.5      | What Is the Model's Output Like? .....           | 219        |
| 5.5.1    | Modeling Behavior .....                          | 220        |
| 5.5.2    | Modeling Experience .....                        | 220        |
| 5.5.3    | No Output .....                                  | 229        |
| 5.6      | How Can We Model Players? .....                  | 230        |
| 5.6.1    | Supervised Learning .....                        | 230        |
| 5.6.2    | Reinforcement Learning .....                     | 235        |
| 5.6.3    | Unsupervised Learning .....                      | 237        |
| 5.7      | What Can We Model? .....                         | 238        |
| 5.7.1    | Player Behavior .....                            | 238        |
| 5.7.2    | Player Experience .....                          | 245        |
| 5.8      | Further Reading .....                            | 252        |
| 5.9      | Exercises .....                                  | 252        |
| 5.9.1    | Player Behavior .....                            | 252        |
| 5.9.2    | Player Experience .....                          | 253        |
| 5.10     | Summary .....                                    | 254        |

**Part III The Road Ahead**

|   |     |
|---|-----|
| <b>6 Game AI Panorama .....</b>                       | 259 |
| 6.1 Panoramic Views of Game AI .....                  | 260 |
| 6.1.1 Methods (Computer) Perspective .....            | 261 |
| 6.1.2 End User (Human) Perspective .....              | 262 |
| 6.1.3 Player-Game Interaction Perspective .....       | 264 |
| 6.2 How Game AI Areas Inform Each Other .....         | 264 |
| 6.2.1 Play Games .....                                | 266 |
| 6.2.2 Generate Content .....                          | 269 |
| 6.2.3 Model Players .....                             | 273 |
| 6.3 The Road Ahead .....                              | 275 |
| 6.4 Summary .....                                     | 277 |
| <b>7 Frontiers of Game AI Research .....</b>          | 279 |
| 7.1 General General Game AI .....                     | 279 |
| 7.1.1 General Play .....                              | 281 |
| 7.1.2 General Game Generation and Orchestration ..... | 282 |
| 7.1.3 General Game Affective Loop .....               | 284 |
| 7.2 AI in Other Roles in Games .....                  | 285 |
| 7.3 Ethical Considerations .....                      | 287 |
| 7.4 Summary .....                                     | 290 |
| <b>References .....</b>                               | 293 |
| <b>Index .....</b>                                    | 331 |



## Acronyms

|        |   |
|--------|---|
| A3C    | Asynchronous Advantage Actor-Critic                           |
| ABL    | A Behavior Language   |
| AFC    | Alternative Forced Choice                                     |
| AI     | Artificial Intelligence                                       |
| AIIDE  | Artificial Intelligence and Interactive Digital Entertainment |
| ALE    | Arcade Learning Environment                                   |
| ANN    | Artificial Neural Network                                     |
| ASP    | Answer Set Programming  |
| BDI    | Belief-Desire-Intention                                       |
| BT     | Behavior Tree   |
| BWAPI  | Brood War API   |
| CA     | Cellular Automata   |
| CI     | Computational Intelligence                                    |
| CIG    | Computational Intelligence and Games                          |
| CFR    | Counterfactual Regret Minimization                            |
| CMA-ES | Covariance Matrix Adaptation Evolution Strategy               |
| CNN    | Convolutional Neural Network                                  |
| CPPN   | Compositional Pattern Producing Network                       |
| DQN    | Deep Q Network  |
| EA     | Evolutionary Algorithm  |
| ECG    | Electrocardiography   |
| EDPCG  | Experience-Driven Procedural Content Generation               |
| EEG    | Electroencephalography  |
| EMG    | Electromyography  |
| FPS    | First-Person Shooter  |
| FSM    | Finite State Machine  |
| FSMC   | Functional Scaffolding for Musical Composition                |
| GA     | Genetic Algorithm   |
| GDC    | Game Developers Conference                                    |
| GGP    | General Game Playing  |
| GSP    | Generalized Sequential Patterns                               |

---

|        |  |
|--------|--|
| GSR    | Galvanic Skin Response                                     |
| GVGAI  | General Video Game Artificial Intelligence                 |
| HCI    | Human-Computer Interaction                                 |
| ID3    | Iterative Dichotomiser 3                                   |
| JPS    | Jump Point Search  |
| LSTM   | Long Short-Term Memory                                     |
| MCTS   | Monte Carlo Tree Search                                    |
| MDP    | Markov Decision Process                                    |
| MLP    | Multi-Layer Perceptron                                     |
| MOBA   | Multiplayer Online Battle Arenas                           |
| NEAT   | NeuroEvolution of Augmenting Topologies                    |
| NES    | Natural Evolution Strategy                                 |
| NLP    | Natural Language Processing                                |
| NPC    | Non-Player Character                                       |
| PC     | Player Character   |
| PCG    | Procedural Content Generation                              |
| PENS   | Player Experience of Need Satisfaction                     |
| PLT    | Preference Learning Toolbox                                |
| RBF    | Radial Basis Function                                      |
| ReLU   | Rectified Linear Unit                                      |
| RPG    | Role-Playing Game  |
| RTS    | Real-Time Strategy   |
| RL     | Reinforcement learning                                     |
| TCIAIG | Transactions on Computational Intelligence and AI in Games |
| TD     | Temporal Difference  |
| ToG    | Transactions on Games                                      |
| TORCS  | The Open Racing Car Simulator                              |
| TRU    | Tomb Raider: Underworld                                    |
| TSP    | Traveling Salesman Problem                                 |
| SC:BW  | StarCraft: Brood War                                       |
| SOM    | Self-Organizing Map  |
| STRIPS | Stanford Research Institute Problem Solver                 |
| SVM    | Support Vector Machine                                     |
| UT2k4  | Unreal Tournament 2004                                     |
| VGDL   | Video Game Description Language                            |

## **Website**

<http://gameaibook.org/>

This book is associated with the above website. The website complements the material covered in the book with up-to-date exercises, lecture slides and readings.



## **Part I**

# **Background**



# Chapter 1

## Introduction

**Artificial Intelligence** (AI) has seen immense progress in recent years. It is both a thriving research field featuring an increasing number of important research areas and a core technology for an increasing number of application areas. In addition to algorithmic innovations, the rapid progress in AI is often attributed to increasing computational power due to hardware advancements. The success stories of AI can be experienced in our daily lives through its many practical applications. AI advances have enabled better understanding of images and speech, emotion detection, self-driving cars, web searching, AI-assisted creative design, and game-playing, among many other tasks; for some of these tasks machines have reached human-level status or beyond.

There is, however, a difference between what machines can do well and what humans are good at. In the early days of AI, researchers envisaged computational systems that would exhibit aspects of human intelligence and achieve human-level problem solving or decision making skills. These problems were presented to the machines as a set of formal mathematical notions within rather narrow and controlled spaces, which could be solved by some form of symbol manipulation or search in symbolic space. The highly formalized, symbolic representation allowed AI to succeed in many cases. Naturally, games—especially **board games**—have been a popular domain for early AI attempts as they are formal and highly constrained, yet complex, decision making environments.

Over the years the focus of much AI research has shifted to tasks that are relatively simple for humans to do but are hard for us to describe how to do, such as remembering a face or recognizing our friend's voice over the phone. As a result, AI researchers began to ask questions such as: *How can AI detect and express emotion?* *How can AI educate people, be creative or artistically novel?* *How can AI play a game it has not seen before?* *How can AI learn from a minimal number of trials?* *How can AI feel guilt?* All these questions pose serious challenges to AI and correspond to tasks that are not easy for us to formalize or define objectively. Perhaps surprisingly (or unsurprisingly after the fact), tasks that require relatively low cognitive effort from us often turn out to be much harder for machines to tackle. Again, games have provided a popular domain to investigate such abilities as they feature

aspects of a subjective nature that cannot be formalized easily. These include, for instance, the experience of play or the creative process of game design [599].

Ever since the birth of the idea of artificial intelligence, **games** have been helping AI research progress. Games not only pose interesting and complex problems for AI to solve—e.g., playing a game well; they also offer a canvas for creativity and expression which is *experienced* by users (people or even machines!). Thus, arguably, games are a rare domain where science (problem solving) meets art and interaction: these ingredients have made games a unique and favorite domain for the study of AI. But it is not only AI that is advanced through games; games have also been advanced through AI research. We argue that AI has been helping games to get better on several fronts: in the way we play them, in the way we understand their inner functionalities, in the way we design them, and in the way we understand play, interaction and creativity. This book is dedicated to all aspects of the intersection of games and AI and the numerous ways both games and AI have been challenged, but nevertheless, advanced through this relationship. It is a book about *AI for games* and *games for AI*.

## 1.1 This Book

The study of AI **in** and **for** games is what this book defines as the research field of **game artificial intelligence** (in brief **game AI**, also occasionally referred to as **AI and games**). The book offers an academic perspective of game AI and serves as a comprehensive guidebook for this exciting and fast-moving research field. Game AI—in particular video game or computer game AI—has seen major advancements in the (roughly) fifteen years of its existence as a separate research field. During this time, the field has seen the establishment and growth of important yearly meetings—including the IEEE Conference on Computational Intelligence and Games (CIG) and the AAAI Artificial Intelligence and Interactive Digital Entertainment (AIIDE) conference series—as well as the launch of the IEEE TRANSACTIONS ON COMPUTATIONAL INTELLIGENCE AND AI IN GAMES (TCIAIG) journal—which will be renamed IEEE TRANSACTIONS ON GAMES (ToG) from January 2018. Since the early days of game AI we have seen numerous success stories within the several subareas of this growing and thriving research field. We can nowadays use AI to play many games better than any human, we can design AI bots that are more believable and human-like than human players, we can collaborate with AI to design better and unconventional (aspects of) games, we can better understand players and play by modeling the overall game experience, we can better understand game design by modeling it as an algorithm, and we can improve game design and tune our monetization strategy by analyzing massive amounts of player data. This book builds on these success stories and the algorithms that took us there by exploring the different **uses** of AI for games and games for AI.

### 1.1.1 Why Did We Write This Book?

Both of us have been teaching and researching game artificial intelligence at undergraduate and graduate levels in various research and educational institutions across the globe for over a decade. Both of us have felt, at times, that a comprehensive **textbook** on game AI was necessary for our students and a service to the learning objectives of our programs. Meanwhile, an increasing number of fellow academics felt the same way. Such a book was not available, and given our extensive experience with the field, we felt we were well placed to write the book we needed. Given that we have been collaborating on game AI research since 2009, and known each other since 2005, we knew our perspective was coherent enough to actually agree on what should go into the book without undue bickering. While we have been trying hard to write a book that will appeal to many and be useful for both students and researchers from different backgrounds, it ultimately reflects our perspective of what game AI is and what is important within the field.

Looking at the existing literature to allocate readings for a potential course on game AI one can rely partly on a small number of relevant and recent surveys and vision papers for specific game AI research areas. Examples include papers meant to serve as general introductions to game AI [407, 764, 785], general game AI [718], Monte-Carlo Tree Search [77], procedural content generation [783, 720], player modeling [782], emotion in games [781], computational narrative [562], AI for game production [564], neuroevolution in games [567], and AI for games on mobile devices [265]. There are also some earlier surveys reflecting the state of the art at the beginning of this research field, for example, on evolutionary computation in games [406] and computational intelligence in games [405]. No mere paper can however on its own cover the breadth and depth required by a full course on game AI. For this reason, the courses we have taught have generally been structured around a set of papers, some of them surveys and some of them primary research papers, together with slides and course notes.

The first, recently published, edited volumes on game AI research have been great assets for teaching needs in game AI. These are books with a focus on a particular area of game AI research such as procedural content generation [616], emotion in games [325] and game data mining [186]. Because of their more narrow domains they cannot serve as textbooks for a complete course on game AI but rather as parts of a game AI course or, alternatively, as textbooks for independent courses on procedural content generation, affective computing or game data mining, for instance.

Meanwhile several edited volumes or monographs which have covered aspects of game AI programming are edited or written by game AI experts from the game industry. These include the popular *game AI programming wisdom* series [546, 547, 548, 549] and other game AI programming volumes [604, 8, 552, 553, 80, 81, 425]. These books, however, target primarily professional or indie developers, game AI programmers and practitioners, and do not always fulfill the requirements of an academic textbook. As you will see, this is only one part of the field of game AI as we define it. Further, some of the earlier books are somewhat outdated by now given the

fast pace of progress of the game AI research field [109, 62]. Among the industry-focused game AI books there are a few that aim to target educators and students of game AI. They have a rather narrow scope, however, as they are limited to non-player character AI [461], which is arguably the most important topic for game AI practitioners in the gaming industry [764, 425] but only one of several research areas in academic game AI research [785]. In our terminology, the perspective of these industry-focused textbooks tends to be almost exclusively on what we call *playing for experience*, in particular generating interesting non-player character (NPC) behavior that looks lifelike and functions within the confines of a game design. Finally there are game AI books that are tied to a particular language or software suite such as Lua [791] or Unity [31], which also limits their usefulness as general textbooks.

In contrast to the above list of books, edited volumes and papers, this book aims to present the research field as a whole and serve (a) as a comprehensive **textbook** for game artificial intelligence, (b) as a **guidebook** for game AI programming, and (c) as a **field guide** for researchers and graduate students seeking to orient themselves within this multifaceted research field. For this reason, we both detail the state of knowledge of the field and also present research and original scholarship in game AI. Thus the book can be used for both research-based teaching and hands-on applications of game AI. We detail our envisaged target audience in the section below.

### 1.1.2 Who Should Read This Book?

With this book we hope to reach readers with a general interest in the application of AI to games, who already know at least the basics of artificial intelligence. However, while writing the book we particularly envisioned three groups of people benefiting directly from this book. The first group is university **students**, of graduate or advanced undergraduate level, who wish to learn about AI in games and use it to develop their career in game AI programming or in game AI research. In particular, we see this book being used in advanced courses for students who have already taken an introductory AI course, but with care and some supplementary material it could be used for an introductory course as well. The second group is AI **researchers** and **educators** who want to use this book to inspire their research or, instead, use it as a textbook for a class in artificial intelligence and games. We particularly think of active researchers within some AI-related field wanting to start doing research in game AI, and new Ph.D. students in the area. The last target audience is computer game **programmers** and practitioners who have limited AI or machine learning background and wish to explore the various creative uses of AI in their game or software application. Here we provide a complement to the more industry-focused books listed above by taking a broader view of what AI in and for games could be. For further fostering the learning process and widening the practical application of AI in games the book is accompanied by a website that features lectures, exercises and additional resources such as readings and tools.

This book is written with the assumption that its readers come from a **technical background** such as computer science, software engineering or applied math. We assume that our readers have taken courses on the fundamentals of artificial intelligence (or acquired this knowledge elsewhere) as the book does not cover the algorithms in detail; our focus, instead, is on the *use* of the algorithms in games and their modification for that purpose. To be more specific, we assume that the reader is familiar with core concepts in tree search, optimization, supervised learning, unsupervised learning and reinforcement learning, and has implemented some basic algorithms from these categories. Chapter 2 provides an overview of core methods for game AI and a refresher for the reader whose knowledge is a bit rusty. We also assume familiarity with programming and a basic understanding of algebra and calculus.

### 1.1.3 A Short Note on Terminology

The term “artificial and computational intelligence in games” is often used to refer to the entire field covered in this book (e.g., see the title of [785]). This reflects the dual roots of the field in artificial intelligence and **computational intelligence** (CI) research, and the use of these terms in the names of the major conferences in the field (AIIDE and CIG) and the flagship journal (IEEE TCIAIG) explicitly targeting both CI and AI research. There is no agreement on the exact meaning of the terms AI and CI. Historically, AI has been associated with logic-based or symbolic methods such as reasoning, knowledge representation and planning, and CI has been associated with biologically-inspired or statistical methods such as neural networks (including what is now known as deep learning) and evolutionary computation. However, there is considerable overlap and strong similarities between these fields. Most of the methods proposed in both fields aim to make computers perform tasks that have at some point been considered to require intelligence to perform, and most of the methods include some form of heuristic search. The field of machine learning intersects with both AI and CI, and many techniques could be said to be part of either field.

In the rest of the book we will use the terms “AI and games”, “AI in games” and “game AI” to refer to the whole research field, including those approaches that originally come from the CI and machine learning fields. There are three reasons for this: simplicity, readability, and that we think that the distinction between CI and AI is not useful for the purposes of this book or indeed the research field it describes. Our use of these terms is not intended to express any prejudice towards particular methods or research questions. (For a non-exclusive list of methods we believe are part of “AI” according to this definition, see Chapter 2.)

## 1.2 A Brief History of Artificial Intelligence and Games

Games and artificial intelligence have a long history together. Much research on AI for games is concerned with constructing agents for playing games, with or without a learning component. Historically, this has been the first and, for a long time, the only approach to using AI in games. Even since before artificial intelligence was recognized as a field, early pioneers of computer science wrote game-playing programs because they wanted to test whether computers could solve tasks that seemed to require “intelligence”. Alan Turing, arguably the principal inventor of computer science, (re)invented the Minimax algorithm and used it to play Chess [725]. The first software that managed to master a game was programmed by A. S. Douglas in 1952 on a digital version of the Tic-Tac-Toe game and as part of his doctoral dissertation at Cambridge. A few years later, Arthur Samuel was the first to invent the form of machine learning that is now called **reinforcement learning** using a program that learned to play Checkers by playing against itself [591].

Most early research on game-playing AI was focused on classic board games, such as Checkers and Chess. There was a conception that these games, where great complexity can arise from simple rules and which had challenged the best human minds for hundreds or even thousands of years, somehow captured the essence of thought. After over three decades of research on tree search, in 1994, the Chinook Checkers player managed to beat the World Checkers Champion Marion Tinsley [594]; the game was eventually solved in 2007 [593]. For decades Chess was seen as “the drosophila of AI” in the sense of being the “model organism” that uncountable new AI methods were tested on [194]—at least until we developed software capable of playing better than humans, at which point Chess-playing AI somehow seemed a less urgent problem. The software that first exhibited superhuman Chess capability, IBM’s Deep Blue, consisted of a Minimax algorithm with numerous Chess-specific modifications and a very highly tuned board evaluation function running on a custom supercomputer [98, 285]. Deep Blue famously won against the reigning grandmaster of Chess, Garry Kasparov, in a much-publicized event back in 1997. Twenty years later, it is possible to download public domain software that will play better than any human player when running on a regular laptop.

A milestone in AI research in games a few years before the successes of Deep Blue and Chinook is the backgammon software named TD-Gammon which was developed by Gerald Tesauro in 1992. TD-Gammon employs an artificial neural network which is trained via temporal difference learning by playing backgammon against itself a few million times [688, 689]. TD-Gammon managed to play backgammon at a level of a top human backgammon player. After Deep Blue IBM’s next success story was Watson, a software system capable of answering questions addressed in natural language. In 2011, Watson competed on the *Jeopardy!* TV game and won \$1 million against former winners of the game [201].

Following the successes of AI in traditional board games the latest board game AI milestone was reached in 2016 in the game of Go. Soon after Chinook and Deep Blue, the game of Go became the new benchmark for game playing AI with a branching factor that approximates 250 and a vast search space many times larger

than that of Chess'. While human level Go playing had been expected sometime in the far future [368], already in 2016 Lee Sedol—a 9-dan professional Go player—lost a five-game match against Google DeepMind's AlphaGo software which featured a deep reinforcement learning approach [629]. Just a few days before the release of the first draft of this book—between 23 and 27 May 2017—AlphaGo won a three-game Go match against the world's number 1 ranking player Ke Jie, running on a single computer. With this victory, Go was the last great classic board game where computers have attained super-human performance. While it is possible to construct classic-style board games that are harder than Go for computers to play, no such games are popular for human players.

But classic board games, with their discrete turn-based mechanics and where the full state of the game is visible to both players, are not the only games in town, and there is more to intelligence than what classic board games can challenge. In the last decade and a half, a research community has therefore grown up around applying AI to games other than board games, in particular video games. A large part of the research in this community focuses on developing AI for **playing games**—either as effectively as possible, or in the style of humans (or a particular human), or with some other property. A notable milestone in video game AI playing was reached in 2014 when algorithms developed by Google DeepMind learned to play several games from the classic Atari 2600 video game console on a super-human skill level just from the raw pixel inputs [464]. One of the Atari games that proved to be hard to play well with that approach was *Ms Pac-Man* (Namco, 1982). The game was practically solved a few days before the release of the second draft of the book (June 2017) by the Microsoft Maluuba team using a hybrid reward architecture reinforcement learning technique [738].

Other uses of AI in video games (as detailed in this book) have come to be very important as well. One of these is **procedural content generation**. Starting in the early 1980s, certain video games created some of their content algorithmically during runtime, rather than having it designed by humans. Two games that became very influential early on are *Rogue* (Toy and Wichmann, 1980), where dungeons and the placement of creatures and items in them are generated every time a new game starts, and *Elite* (Acornsoft, 1984), which stores a large universe as a set of random seeds and creates star systems as the game is played. The great promise of games that can generate some of their own content is that you can get more—potentially infinite—content without having to design it by hand, but it can also help reduce storage space demands among many other potential benefits. The influence of these games can be seen in recent successes such as *Diablo III* (Blizzard Entertainment, 2012), *No Man's Sky* (Hello Games, 2016) and the Chalice Dungeons of *Bloodborne* (Sony Computer Entertainment, 2015).

Relatively recently, AI has also begun to be used to analyze games, and **model players** of games. This is becoming increasingly important as game developers need to create games that can appeal to diverse audiences, and increasingly relevant as most games now benefit from internet connectivity and can “phone home” to the developer's servers. Facebook games such as *FarmVille* (Zynga, 2009) were among the first to benefit from continuous data collection, AI-supported analysis of the data

and semi-automatic adaptation of the game. Nowadays, games such as *Nevermind* (Flying Mollusk, 2016) can track the emotional changes of the player and adapt the game accordingly.

Very recently, research on believable agents in games has opened new horizons in game AI. One way of conceptualizing believability is to make agents that can pass game-based Turing tests. A game Turing test is a variant of the Turing test in which a number of judges must correctly guess whether an observed playing behavior in a game is that of a human or an AI-controlled game bot [263, 619]. Most notably, two AI-controlled bot entries managed to pass the game Turing test in *Unreal Tournament 2004* (Epic Games, 2004) on Turing’s centenary in 2012 [603].

In the next section we will outline the parallel developments in both academia and industry and conclude the historical section on game AI with ways the two communities managed to exchange practices and transfer knowledge for a common two-fold goal: the advancement of AI and the improvement of games at large.

### 1.2.1 Academia

In academic game AI we distinguish two main domains and corresponding research activity: board games and video (or computer) games. Below, we outline the two domains in a chronological sequence, even though game AI research is highly active in both of them.

#### 1.2.1.1 Early Days on the Board

When it comes to game AI research, classic board games such as Chess, Checkers and Go are clearly beneficial to work with as they are very simple to model in code and can be simulated extremely fast—one can easily make millions of moves per second on a modern computer—which is indispensable for many AI techniques. Also, board games seem to require thinking to play well, and have the property that they take “a minute to learn, but a lifetime to master”. It is indeed the case that games have a lot to do with learning, and good games are able to constantly teach us more about how to play them. Indeed, to some extent the fun in playing a game consists in learning it and when there is nothing more to learn we largely stop enjoying them [351]. This suggests that better-designed games are also better benchmarks for artificial intelligence. As mentioned above, board games were the dominant domain for AI research from the early 1950s until quite recently. As we will see in the other parts of this book—notably in Chapter 3—board games remain a popular game AI research domain even though the arrival of video and arcade games in the 1980s has shifted a large part of the focus since then.

### 1.2.1.2 The Digital Era

To the best of our knowledge, the first *video* game conference occurred at Harvard’s Graduate School of Education<sup>1</sup> in 1983. The core focus of the conference was on the educational benefits and positive social impact of video game playing.

The birth date of the digital game AI field can be safely considered to be around year 2001. The seminal article by Laird and van Lent [360], emphasizing the role of games as the **killer application** for AI, established the foundations of game AI and inspired early work in the field [696, 235, 476, 292, 211, 694, 439, 766, 707]. In those early days AI in digital games was mainly concerned with playing games, agent architectures for NPC behavior [401, 109], sometimes within interactive drama [438, 399, 412, 483, 107], and pathfinding [664]. Early work in these areas was presented primarily in the AAAI Spring Symposia on AI and Interactive Entertainment preceding the AIIDE (which started 2005) and the IEEE CIG (also started in 2005) conferences. Most of the early work in the game AI field was conducted by researchers with AI, optimization and control background and research experience in adaptive behavior, robotics and multi-agent systems. AI academics used the best of their computational intelligence and AI tools to enhance NPC behavior in generally simple, research-focused, non-scalable projects of low commercial value and perspective.

### 1.2.2 Industry

The first released video games back in the 1970s included little or nothing that we would call artificial intelligence; NPC behaviors were scripted or relied on simple rules, partly because of the primitive state of AI research, but perhaps even more because of the primitive hardware of the time. However, in parallel to developments in academia, the game industry gradually made steps towards integrating more sophisticated AI in their games during the early days of game AI [109, 758].

A non-exhaustive list of AI methods and game features that advanced the game AI state-of-practice in the industry [546] in chronological order includes the first popular application of neural networks in *Creatures* (Millennium Interactive, 1996) with the aim to model the creatures’ behavior; the advanced sensory system of guards in *Thief* (EIDOS, 1998); the team tactics and believable combat scenes in the *Halo* series (Microsoft Studios, 2011–2017)—*Halo 2* in particular popularized the use of behavior trees in games; the behavior-based AI of *Blade Runner* (Virgin Interactive, 1997); the advanced opponent tactics in *Half-Life* (Valve, 1998); the fusion of machine learning techniques such as perceptrons, decision trees and reinforcement learning coupled with the belief-desire-intention cognitive model in *Black and White* (EA, 2000)—see Fig. 1.1; the believable agents of *The Sims* series (Electronic Arts, 2000–2017); the imitation learning *Drivatar* system of *Forza Motorsport* (MS

---

<sup>1</sup> Fox Butterfield, *Video Game Specialists Come to Harvard to Praise Pac-Man; Not to Bury Him*. New York Times, May 24, 1983



**Fig. 1.1** A screenshot from *Black and White* (EA, 2000), a highlight in game artificial intelligence history that successfully integrated several AI methods into its design. The game features a creature that learns through positive rewards and penalties in a reinforcement learning fashion. Further, the creature employs the belief-desire-intention model [224] for its decision making process during the game. The desires of the creature about particular goals are modeled via simple perceptrons. For each desire, the creature selects the belief that it has formed the best opinion about; opinions, in turn, are represented by decision trees. Image obtained from Wikipedia (fair use).

Game Studios, 2005); the generation of context-sensitive behaviors via Goal Oriented Action Planning [506]—a simplified STRIPS-like planning method—which was specifically designed for *F.E.A.R.* (Sierra Entertainment, 2005) [507]; the procedurally generated worlds of the *Civilization* series (MicroProse, Activision, Infogrames Entertainment, SA and 2K Games, 1991–2016) and *Dwarf Fortress* (Bay 12 Games, 2006); the AI director of *Left 4 Dead* (Valve, 2008); the realistic gunfights of *Red Dead Redemption* (Rockstar Games, 2010); the personality-based adaptation in *Silent Hill: Shattered Memories* (Konami, 2010); the affect-based cinematographic representation of multiple cameras in *Heavy Rain* (Quantic Dream, 2010); the neuroevolutionary training of platoons in *Supreme Commander 2* (Square Enix, 2010); the buddy AI (named Ellie) in *The Last of Us* (Sony Computer Entertainment, 2013); the companion character, Elizabeth, in *BioShock Infinite* (2K Games, 2013); the interactive narratives of *Blood & Laurels* (Emily Short, 2014); the alien’s adaptive behavior which adjusts its hunting strategy according to the player in *Alien: Isolation* (Sega, 2014); and the procedurally generated worlds of *Spelunky* (Mossmouth, LLC, 2013) and *No Man’s Sky* (Hello Games, 2016).

The key criterion that distinguishes successful AI in commercial-standard games had always been the level of integration and interweaving of AI in the design of the game [599, 546]. An unsuccessful coupling of game design and AI may lead to unjustifiable NPC behaviors, break the suspension of disbelief and immediately reduce player immersion. A typical example of such a mismatch between AI and design is the broken navigation of bots that get stuck in a level’s dead end; in such instances either the level design is not (re)considered appropriately to match the design of AI or the AI is not tested sufficiently, or both. On the other hand, the successful integration of AI in the design process is likely to guarantee satisfactory outcomes for the playing experience. The character design process, for instance, may consider the limitations of the AI and, in turn, absorb potential “catastrophic” failures of it. An example of such an interwoven process is the character design in *Façade* [441] that was driven, in part, by the limitations of the natural language processing and the interactive narrative components of the game.

It is important to note that this book is not necessarily about game AI as defined and practiced in the game industry. Instead, it is primarily an **academic textbook** that refers to some of the techniques that have been used in and popularized through the game industry—see for instance the *ad-hoc behavior authoring* section of Chapter 2. The reader with an interest in the AI state of practice in the game industry is referred to the several introductory articles (e.g., [171, 369]) available in books such as the *game AI programming wisdom* series [546, 547, 548, 549]. Another valuable resource is the video recorded talks from top game AI programmers which are hosted at the AI summit<sup>2</sup> of the Game Developers Conference (GDC) and are available at the GDC Vault.<sup>3</sup> Finally, talks and courses mostly relevant for game AI programmers are available through the nucl.ai conference webpage.<sup>4</sup>

### 1.2.3 The “Gap”

During the first decade of academic game AI research, whenever researchers from academia and developers from industry would meet and discuss their respective work, they would arrive at the conclusion that there exists a *gap* between them; the gap had multiple facets such as differences in background knowledge, practice, trends, and state-of-the-art solutions to important problems. Academics and practitioners would discuss ways to bridge that gap for their mutual benefit on a frequent basis [109] but that debate would persist for many years as developments on both ends were slow. The key message from academic AI was that the game industry should adopt a “high risk-high gain” business model and attempt to use sophisticated AI techniques with high potential in their games. On the other end, the central complaint of industrial game AI regarding game AI academics has been the lack of

<sup>2</sup> <http://www.gdconf.com/conference/ai.html>

<sup>3</sup> <http://www.gdcvault.com/>

<sup>4</sup> <https://nucl.ai/>

domain-specific knowledge and practical wisdom when it comes to realistic problems and challenges faced during game production. Perhaps above all there is a difference in what is valued, with academics valuing new algorithms and new uses of algorithms that achieve superior performance or create new phenomena or experiences, and AI developers in industry valuing software architectures and algorithmic modifications that reliably support specific game designs. But what happened since then? Does this gap really exist nowadays or is it merely a ghost from the past?

It is still true that the academic game AI research community and the game industry AI development community largely work on different problems, using different methods. There are also some topics and methods explored by the academic community which are generally very unpopular within the game industry. Real-time adaptation and learning in NPCs is one such example; quite a few academic researchers are excited by the idea of NPCs that can learn and develop from their interactions with the player and other NPCs in the game. However, AI developers in industry point out that it would be very hard to predict what these NPCs will learn, and it is very likely to “break the game” in the sense that it no longer works as designed. Conversely, there are methods and problems explored in industry which most academics do not care about, as they only make sense within the complex software architecture of a complete game.

When thinking about the use of AI within modern video games, it is important to remember that most game genres have developed evolutionarily from earlier game designs. For example, the first platformers were released in the mid-1980s and the first first-person shooters and real-time strategy games in the early 1990s. At that time, the ability to build and deploy advanced AI was much less than it is today, so designers had to design around the lack of AI. These basic design patterns have largely been inherited by today’s games. It can therefore be said that many games have been designed not to need AI. For the academic who wants to build interesting AI for an in-game role, the best might therefore be to create new game designs that start from the existence of the AI.

Taking a positive stance on the topic we would argue that any existing gap between academic and industrial NPC AI nowadays can be viewed as a healthy indication of a parallel progress with a certain degree of collaboration. As industry and academia do not necessarily attempt to solve the same problems with the same approaches it may be that NPC AI solutions emerging from industry can inspire new approaches in academia and vice versa. In summary, the NPC AI gap is clearly smaller in tasks that both academia and industry care about. Certain aspects of NPC AI, however, are far from being solved in an ideal fashion and others—such as modeling emotion in role playing games—are still at the beginning stages of investigation. So while we can praise the NPC AI of *The Elder Scrolls V: Skyrim* (Bethesda Softworks, 2011) we cannot be as positive about the companion AI of that game. We can view the very existence of such limitations as an opportunity that can bring industry and academia even closer to work on further improving existing NPC AI in games.

A different take on this discussion—which is supported by some game developers and game AI academics—is that NPC AI is almost solved for most production

tasks; some take it one step further and claim that game AI research and development should focus solely on non-traditional uses of AI [477, 671]. The level of AI sophistication in recent games such as *Left 4 Dead* (Valve, 2008) and *The Elder Scrolls V: Skyrim* (Bethesda Softworks, 2011) supports this argument and suggests that advances in NPC AI have reached satisfactory levels for many NPC control challenges faced during game production. Due to the rise of robust and effective industrial game AI solutions, the convergence to satisfying NPC performances, the support of the multidisciplinary nature of game AI and a more pragmatic and holistic view of the game AI problem, recent years have seen a shift of academic and industrial interests with respect to game AI. It seems that we have long reached an era where the primary focus of the application of AI in the domain of games is not on agents and NPC behaviors. The focus has, instead, started to shift towards interweaving game design and game technology by viewing the role of AI holistically and integrating aspects of procedural content generation and player modeling within the very notion of game AI [764].

The view we take in this book is that AI can help us to make better games but that this does not necessarily happen through better, more human-like or believable NPCs [764]. Notable examples of non-NPC AI in games include *No Man's Sky* (Hello Games, 2016) and its procedural generation of a quintillion different planets and *Nevermind* (Flying Mollusk, 2016) with its affective-based game adaptation via a multitude of physiological sensors. But there might be other AI roles with game design and game development that are still to be found by AI. Beyond playing games, modeling players or generating content, AI might be able to play the role of a design assistant, a data analyst, a playtester, a game critic, or even a game director. Finally, AI could potentially play and design games as well as model their players in a general fashion. The final chapter (Chapter 7) of this book is dedicated to these frontier research areas for game AI.

## 1.3 Why Games for Artificial Intelligence

There are a number of reasons why games offer the ideal domain for the study of artificial intelligence. In this section, we list the most important of them.

### 1.3.1 Games Are Hard and Interesting Problems

Games are engaging due to the effort and skills required from people to complete them or, in the case of puzzles, solve them. It is that *complexity* and *interestingness* of games as a problem that makes them desirable for AI. Games are **hard** because their finite state spaces, such as the possible strategies for an agent, are often vast. Their complexity as a domain rises as their vast search spaces often feature small

feasible spaces (solution spaces). Further, it is often the case that the goodness of any game state is hard (or even impossible) to assess properly.

From a computational complexity perspective, many games are NP-hard (NP refers to “nondeterministic polynomial time”), meaning that the worst-case complexity of “solving” them is very high. In other words, in the general case an algorithm for solving a particular game could run for a very long time. Depending on a game’s properties complexity can vary substantially. Nevertheless, the list of games that are NP-hard is rather long and includes games such as the two-player incomplete information *Mastermind* game [733, 660], the *Lemmings* (Psygnosis, 1991) arcade game [334] and the *Minesweeper* game by Microsoft [331]. It should be noted that this computational complexity characterization has little to do with how hard the games are to play for humans, and does not necessarily say much about how well heuristic AI methods can play them. However, it is clear that at least in theory, and for arbitrary-size instances, many games are very hard.

The investigations of AI capacity in playing games that are hard and complex has been benchmarked via a number of milestone games. As mentioned earlier, Chess and (to a lesser degree) Checkers have traditionally been seen as the “drosophila for AI research” even from the early days of AI. After the success of Deep Blue and Chinook in these two games we gradually invented and cited other more complex games as AI “drosophilae”, or universal benchmarks. *Lemmings* has been characterized as such; according to McCarthy [446] it “connects logical formalizations with information that is incompletely formalizable in practice”. In practice, games for which better APIs have been developed—such as *Super Mario Bros* (Nintendo, 1985)<sup>5</sup> and *StarCraft* (Blizzard Entertainment, 1998)—have become more popular benchmarks.

The game of computer Go has also been another core and traditional game AI benchmark with decades of active research. As a measure of problem complexity a typical game in Go has about  $10^{170}$  states. The first AI feature extraction investigations in Go seem to date back to the 1970s [798]. The game received a lot of research attention during several world computer Go championships up until the recent success of AlphaGo [629]. AlphaGo managed to beat two of the best Go professional human players using a combination of deep learning and Monte Carlo tree search. In March 2016, AlphaGo won against Lee Sedol and in May 2017 it won all three games against the world’s number 1 ranked player, Ke Jie.

The *StarCraft* (Blizzard Entertainment, 1998) real-time strategy game can be characterized as perhaps the single hardest game for computers to play well. At the time of writing this book the best *StarCraft* bots only reach the level of amateur players.<sup>6</sup> The complexity of the game derives mainly from the multi-objective task of controlling multiple and dissimilar units in a game environment of partial information. While it is not trivial to approximate the state space of *StarCraft*, according to a recent study [729], a typical game has at least  $10^{1,685}$  possible states. In comparison, the number of protons in the observable universe is only about  $10^{80}$  [182].

---

<sup>5</sup> Note that the original game title contains a dot, i.e., *Super Mario Bros.*; for practical reasons, however, we will omit the dot when referring to the game in the remainder of this book.

<sup>6</sup> <http://www.cs.mun.ca/~dchurchill/starcrafttaicomp/>

The number of *StarCraft*'s possible states sounds huge but, interestingly enough, its search space can be of manageable size if represented by bytes. On that basis, we require about 700 bytes of information to represent the *StarCraft* search space whereas the number of protons in the known universe is equivalent to the number of configurations of about 34 bytes.

One could of course design games that are harder on purpose, but there is no guarantee anyone would want to play those games. When doing AI research, working on games that people care about means you are working on relevant problems. This is because games are designed to challenge the human brain and successful games are typically good at this. *StarCraft* (Blizzard Entertainment, 1998)—and its successor *StarCraft II* (Blizzard Entertainment, 2010)—are played by millions of people all over the world, with a very active competition scene of elite professional players and even dedicated TV channels such as OGN<sup>7</sup>—a South Korean cable television channel—or twitch channels<sup>8</sup> that specialize in broadcasting video game-related content and e-sports events.

Many would claim that *StarCraft* (Blizzard Entertainment, 1998) is the next major target for AI research on playing to win. In academia, there is already a rich body of work on algorithms for playing (parts of) *StarCraft* [504, 569, 505, 124], or generating maps for it [712]. Beyond academia, industrial AI leaders Google DeepMind and Facebook seem to be in agreement and on a similar scientific mission. DeepMind recently announced that *StarCraft II* will be one of their major new testbeds, after their success at training deep networks to play Atari games in the Arcade Learning Environment<sup>9</sup> (ALE) [40] framework. At the time of writing this book, DeepMind in collaboration with Blizzard Entertainment opened up *StarCraft II* to AI researchers for testing their algorithms.<sup>10</sup> Facebook AI Research has led the development of *TorchCraft* [681]—a bridge between the deep learning Torch library and *StarCraft*—and recently published their first paper on using machine learning to learn to play *StarCraft* [729], showing that they take this challenge seriously. Another industrial game AI research lab collaborating with academia on solving *StarCraft* is hosted in Alibaba [523]. Given the game's complexity, it is unlikely we will conquer all of it soon [234] but it is a game through which we expect to see AI advancements in the years to come.

### 1.3.2 Rich Human-Computer Interaction

Computer games are dynamic media by definition and, arguably, offer one of the **richest** forms of human-computer interaction (HCI); at least at the time of writing this book. The richness of interaction is defined in terms of the available options

---

<sup>7</sup> <http://ch.interest.me/ongamenet/>

<sup>8</sup> For instance, <https://www.twitch.tv/starcraft>

<sup>9</sup> <http://www.arcadelearningenvironment.org/>

<sup>10</sup> Follow developments at: <https://deepmind.com/blog/>

a player has at any given moment and the ways (modalities) a player can interact with the medium. The available options for the player are linked to the game action space and the complexity associated with it in games such as *StarCraft II* (Blizzard Entertainment, 2010). Further, the modalities one may use to interact with games nowadays extend beyond the traditional keyboard, mouse and tablet-like haptics, to game controllers, physiology such as heart rate variability, body movement such as body stance and gestures, text, and speech. As a result, many games would easily top the list of information bits exchanged between them and their users per second compared to any other HCI medium; however, such comparative studies are not currently available to further support our claims.

Clearly, as we will see later in this book, games offer one of the best and most meaningful domains for the realization of the **affective loop**, which defines a framework that is able to successfully elicit, detect and respond to the cognitive, behavioral and emotive patterns of its user [670]. The potential that games have to influence players is mainly due to their ability to place the player in a continuous mode of interaction with the game which elicits complex cognitive, affective and behavioral responses to the player. This continuous interaction mode is enriched by fast-paced and multimodal forms of user interactivity that are often possible in games. As every game features a player—or a number of players—the interaction between the player and the game is of key importance for AI research as it gives algorithms access to rich player experience stimuli and player emotional manifestations. Such complex manifestations, however, cannot trivially be captured by standard methods in machine learning and data science. Undoubtedly, the study of game-player interaction via artificial intelligence not only advances our knowledge about human behavior and emotion but also contributes to the design of better human-computer interaction. As a result, it further pushes the boundaries of AI methods in order to address the challenges of game-based interaction.

### 1.3.3 Games Are Popular

While video games, back in the 1980s, were introduced as a niche activity for those having access to a video arcade or to consoles such as Atari 2600, they gradually turned into a multi-billion industry generating, in the 2010s, a global market revenue higher than any other form of creative industry, including film and music. At the time of writing, games generate a worldwide total of almost \$100 billion in revenue which is expected to rise to approximately \$120 billion by 2019.<sup>11</sup>

But why did games become so popular? Beyond the obvious argument of games being able to enhance a user's intrinsic motivation and engagement by offering interactivity capacities with a virtual environment, it was the technological advancements over the last 40 years that drastically changed the demographics of players [314]. Back in the early 1980s games used to be played solely in arcade entertain-

---

<sup>11</sup> See, for instance, the global games market report by Newzoo:  
<https://newzoo.com/solutions/revenues-projections/global-games-market-report/>

ment machines; nowadays, however, they can be played using a multitude of devices including a PC (e.g., multi-player online or casual games), a mobile phone, a tablet, a handheld device, a virtual reality device, or a console (and obviously still an arcade entertainment machine!). Beyond the technological advancements that fostered accessibility and democratized gameplay, it is also the culture that follows a new medium and develops it into a new form of art and expression. Not only the independent scene of game design and development<sup>12</sup> has contributed to this culture, but also the outreach achieved from the multitude of purposes and objectives of games beyond mere entertainment: games for art, games as art, games for a change, physical interactive games, games for education, games for training and health, games for scientific discovery, and games for culture and museums. In brief, not only are games everywhere and present in our daily lives but they also shape our social and cultural values at large—as for instance evidenced by the recent massive success of *Pokémon Go* (Niantic, 2016). As a byproduct of their popularity games offer easy access to people with world-class performance in the domain. Experts (i.e., professional players) for many board and digital games that are world-ranked according to their gameplay performance have participated regularly in competitions against AI algorithms; examples include Garry Kasparov (Chess), and Lee Sedol and Ke Jie (Go).

As games become more popular, grow in quantity, and become more complex, new AI solutions are constantly required to meet the new technological challenges. This is where AI meets a domain with a strong industrial backing and a desire to support sophisticated technology for bettering player experience. Furthermore, very few domains for AI offer the privilege of daily accessibility to new **content** and **data** from their popular use. But let us look at these two aspects in more detail below.

### 1.3.3.1 Popular Means More Content

The more people play (more) games, the more content is required for games. Content takes effort to create but, over the years, mechanisms have been developed that allow both machines and players to design and create various forms of content in games. Games have gradually developed to be **content-intensive** software applications that demand content that is both of direct use in the game and of sufficient novelty. The overwhelming demand for new and novel gaming experiences from a massive community of users constantly pushes the boundaries of human and computational creativity to new grounds; and naturally AI at large.

Content in games, beyond any other form of multimedia or software application, not only covers all possible forms of digital content such as audio, video, image, and text but it also comes in massive numbers of different resolutions and representations. Any algorithm that attempts to retrieve and process the variety and amount of content within or across games is directly faced with the challenges of interoperability and content convergence as well as scalability caused by such big data sets.

---

<sup>12</sup> <http://www.igf.com/>

Compare this with a typical robot simulator, where all the environments would have to be painstakingly hand-crafted or adapted from data collected from the real world. When using games as testbeds, there is no such content shortage.

### 1.3.3.2 Popular Means More Data

Massive content creation (either by games or players) is one major effect of games' popularity; the other is massive data generation of game playthroughs and player behavior. Since the late 2000s, game companies have had access to accurate game telemetry services that allow them to track and monitor player purchases, churn and re-engagement, or the progress of play for debugging either the game or the players' experience. The algorithmic challenges met here follow the general challenges of big data and big data mining research [445], which include data filtering during data acquisition, metadata generation, information extraction from erroneous and missing data, automatic data analysis across dissimilar datasets, appropriate declarative query and mining interfaces, scalable mining algorithms, and data visualization [359]. Luckily enough some of these datasets are nowadays openly available for game analytics and game data mining research. Indicatively, in March 2017, the OpenDota project<sup>13</sup>—a community-maintained open source Dota 2 data platform—released a sanitized archive of over a **billion matches** (!) of *Dota 2* (Valve Corporation, 2013) that were played between March 2011 and March 2016.<sup>14</sup>

### 1.3.4 There Are Challenges for All AI Areas

Unlike some more narrow benchmarks, games challenge *all* core areas of AI. This can be seen by taking a number of broadly accepted areas of AI and discussing the challenges available for those areas in games. **Signal processing**, for starters, meets great challenges in games. Data from players, for instance, not only come in different resolutions—in-game events vs. head pose vs. the player's physiology—they also originate from multiple modalities of fast-paced interaction in an environment that elicits complex cognitive and affective patterns to the player. Multi-modal interaction and multi-modal fusion are non-trivial problems when building embodied conversational agents and virtual characters. Further, the complexity of the signal processing task in games is augmented due to the spatio-temporal nature of the signals which is caused by the rich and fast-paced interaction with the game.

As discussed in the introductory section of this chapter Checkers, Chess, Jeopardy!, Go and arcade games mark a historical trace of core major milestones for **machine learning** (Go and arcade games), **tree search** (Checkers and Chess), **knowledge representation and reasoning** (Jeopardy!) and **natural language processing**

---

<sup>13</sup> <https://www.opendota.com/>

<sup>14</sup> <https://blog.opendota.com/2017/03/24/datadump2/>

(Jeopardy!, Kinect games) and, in turn, they resulted in major breakthroughs for AI. This historical association between AI accomplishments and games already provides clear evidence that all the above areas have traditionally been challenged by games. While the full potential of machine learning remains to be discovered in games such as *StarCraft II* (Blizzard Entertainment, 2010), natural language processing (NLP) has been challenged deeply through games involving narrative and natural language input. NLP is challenged even further in game environments that wish to realize forms of interactive storytelling [148].

Finally when it comes to **planning** and **navigation**, games have traditionally offered environments of high and increasing complexity for algorithms to be tested in. While games such as *StarCraft* clearly define major milestones for planning algorithms, navigation and pathfinding have reached a certain degree of maturity through simulated and roborealistic game environments featuring multiple entities (agents). An additional benefit of games as a domain for behavioral planning is that they offer a realistic yet a far more convenient and cheaper testbed compared to robotics. Beyond the extensive testing and advancement of variants of A\* through games, popular and highly effective tree search variants such as the Monte Carlo tree search [77] algorithm have been invented in response to problems posed by game-playing.

### 1.3.5 Games Best Realize Long-Term Goals of AI

One of the long-standing questions that AI is faced with is *what is the ultimate long-term goal for AI?* While numerous debates and books have been dedicated to this topic, the collaborative effort of Wikipedia authors addressing this question<sup>15</sup> reveals the areas of **social intelligence** and **affective interaction**, (computational) **creativity**, and **general intelligence** as the most critical long-term goals of AI. Our reference has been critiqued for systemic bias (in any controversial question such as the one above); we argue, however, that any reference on this topic would be subjective anyhow. Without aiming to be exclusive or biased, we believe that the three aforementioned areas collectively contribute to better AI systems and we discuss why games best realize these three goals below. These three long-term goals define frontier research areas for game AI and are further elaborated on in the last chapter of this book.

#### 1.3.5.1 Social and Emotional Intelligence

Affective computing [530] is the multidisciplinary area of study across computer science, cognitive science and psychology that investigates the design and development of intelligent software that is able to elicit, detect, model, and express emotion

---

<sup>15</sup> Wikipedia (accessed: May 2017): [https://en.wikipedia.org/wiki/Artificial\\_intelligence](https://en.wikipedia.org/wiki/Artificial_intelligence)

and social intelligence. The ultimate aim of affective computing is the realization of the so-called *affective loop* [670] which, as we covered earlier, defines a system that is able to successfully elicit, detect and respond to the emotions of its user. Naturally, both emotive and social aspects of intelligence are relevant for a system that realizes the affective loop.

Games can offer a highly meaningful realization of the affective loop and affective interaction [781]. Games are by definition both *entertaining* (whether used for pure satisfaction, training or education) and *interactive* activities that are played within fantasy worlds. Thus, any limitations of affective interaction—such as the difficulty to justify affective-based game decisions or alterations of content to the user—are absorbed naturally. For example, an erroneous affective response of a game character can still be justified if the design of the character and the game context do not break the *suspension of disbelief* of the player—during which the player ignores the medium and the interaction for the sake of her enjoyment. Further, games are designed to offer affective experiences which are influenced by player feedback and players are willing to go through, e.g., frustrating, anxious, and fearful episodes of play for experiencing involvement. To that end, a user under gaming conditions—more than any other form of human-computer interaction—is generally open to affective-based alterations of the interaction and influences of his/her emotional state.

### 1.3.5.2 Computational Creativity

Computational creativity studies the potential of software to autonomously generate outcomes that can be considered creative or algorithmic processes that are deemed to be creative [54, 754]. Computer games can be viewed as the killer application domain for computational creativity [381]. It is not only their unique features that we covered in earlier sections of this chapter—i.e., being highly interactive, dynamic and content-intensive software applications. Most importantly it is their *multifaceted* nature. In particular, it is the fusion of the numerous and highly diverse creative domains—visual art, sound design, graphic design, interaction design, narrative, virtual cinematography, aesthetics and environment beautification—with a single software application that makes games the ideal arena for the study of computational creativity. It is also important to note that each art form (or facet) met in games elicits different experiences to its users; their fusion in the final software targeting a rather large and diverse audience is an additional challenge for computational creativity.

As a result, the study of computational creativity *within* and *for* computer games [381] advances in both the field of AI and the domain of games. Games can, first, be improved as products via computational creations (*for*) and/or, second, be used as the ultimate canvas for the study of computational creativity as a process (*within*). Computer games not only challenge computational creativity but they also provide a creative sandbox for advancing the field. Finally, games can offer an opportunity for computational creativity methods to be extensively assessed via a huge population of users of commercial-standard products of high impact and financial value.

### 1.3.5.3 General Intelligence

AI has investigated the general intelligence capacity of machines within the domain of games more than any other domain thanks to the ideal properties of games for that purpose: controlled yet interesting and computationally hard problems [598]. In particular, the capacity of AI to play unseen games well—i.e., general game playing—has seen a number of advancements in recent years. Starting with the general game playing competition [223], focusing on board games and similar discrete perfect information games, we now also have the Arcade Learning Environment [40] and the General Video Game AI Competition [528], which offer radically different takes on arcade video games. Advancements vary from the efforts to create game description languages suitable for describing games used for general game playing [533, 223, 400, 691, 354, 596, 181, 429] to the establishment of a set of general video game AI benchmarks [223, 528, 40] to the recent success of deep Q-learning in playing arcade games with human-level performance just by processing the screen’s pixels [464].

While general game playing is studied extensively and constitutes one of the key areas of game AI [785], we argue that the focus of generality solely with regard to the performance of game-playing agents is *very narrow* with respect to the spectrum of roles for general intelligence in games. The types of general intelligence required within game development include game and level design as well as player behavior and experience modeling. Such skills touch upon a diverse set of cognitive and affective processes which have until now been ignored by general AI in games. For general game AI to be truly general and advance AI algorithmically, it needs to go beyond game playing while retaining the focus on addressing more than a single game or player [718]. We further argue that the challenge of bringing together different types of skillsets and forms of intelligence within autonomous designers of games cannot only advance our knowledge about human intelligence but also advance the capacity of general artificial intelligence.

## 1.4 Why Artificial Intelligence for Games

The various uses of AI in games are beneficial for the design of better games for a number of reasons. In this section we focus on the benefits obtained by allowing AI to play a game, to generate content and to analyze player experience and behavior.

### 1.4.1 AI Plays and Improves Your Game

AI can improve games in several ways by merely playing them. The game industry usually receives praise for the AI of their games—in particular, the non-player or opponent AI—when the AI of the game adds to the commercial value of the game,

it contributes to better game reviews, and it enhances the experience of the player. Whether the underlying AI is based on a simple behavior tree, a utility-based AI or alternatively on a sophisticated machine learned reactive controller is of limited relevance as long as it serves the aforementioned purposes. An unconventional and effective solution to an NPC task can often be a critical factor that shapes management, marketing and monetization strategies during and after production.

As we will see in Chapter 3, AI plays games with two core objectives in mind: play **well** and/or play **believably** (or human-like, or interestingly). Further AI can control either the **player** character or the **non-player** character of the game. AI that plays well as a player character focuses on optimizing the *performance* of play—performance is measured as the degree to which a player meets the objectives of the game solely. Such AI can be of tremendous importance for automatic game testing and for the evaluation of the game design as a whole. AI that plays well as a non-player character, instead, can empower **dynamic difficulty adjustment** and automatic game balancing mechanisms that will in turn personalize and enhance the experience for the player (as in [651] among many). If the focus of AI is shifted on controlling player characters that play believably or human-like (as in [96, 719, 264] among many) then AI can serve as means for player experience debugging or as demonstration of realistic play for design purposes. Finally, a game that features a rich interaction with NPCs can only benefit from AI that controls NPCs which are expressive and depict human-like and believable behaviors (as in [563, 683, 762] among many).

### 1.4.2 More Content, Better Content

There are several reasons for game designers and developers to be interested in AI and, in particular, in content generation as covered in detail in Chapter 4. The first and most historical reason is **memory consumption**. Content can typically be compressed by keeping it “unexpanded” until needed. A good example is the classic space trading and adventure game *Elite* (Acornsoft, 1984), which managed to keep hundreds of star systems in a few tens of kilobytes of memory available on the hardware. Further, content generation might foster or further inspire human **creativity** and allow the emergence of completely new types of games, game genres or entirely new spaces of exploration and artistic expression [381]. Moreover, if new content can be generated with sufficient variety, quality and quantity, then it may become possible to create truly endless games with **ultimate replay** value. Finally, when content generation is associated with aspects of play we can expect personalized and **adaptive play** to emerge via the modification of content.

Unlike other areas of game AI—such as general game playing which might be considered more of an academic pursuit—content generation is a commercial necessity [381]. Prior to the academic interest in content generation—which is rather recent [704, 720, 616]—content generation systems had a long history of supporting commercial standard games for creating engaging yet unpredictable game experi-

ences but, most importantly, lessening the burden of manual game content creation. Naturally, games that feature sophisticated content generation systems can garner praise for their technologies—as in *Diablo III* (Blizzard, 2012)—or even build an entire marketing campaign on content generation—like *No Man’s Sky* (Hello Games, 2016).

### 1.4.3 Player Experience and Behavioral Data Analytics

The use of AI for the understanding of player experience can drive and enhance the design process of games. Game designers usually explore and test a palette of mechanics and game dynamics that yield experience patterns they desire to put the player through. Player states such as engagement, fear and stress, frustration, anticipation, and challenge define critical aspects of the design of player experience, which is dependent on the genre, the narrative and the objectives of the game. As a result, the holy grail of game design—that is player experience—can be improved and tailored to each player but also augmented via richer experience-based interaction. Further, as a direct consequence of better and faster design, the whole game development process is boosted and improved.

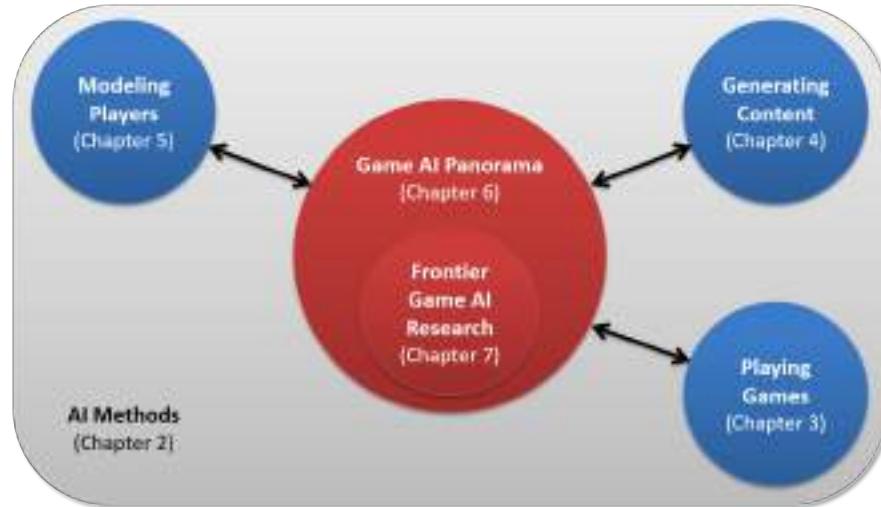
Beyond the experience of the player, data derived from games, their use and their players provide a new and complementary way of designing games, of making managerial and marketing decisions about games, of affecting the game production, and of offering a better customer service [178]. Any AI-informed decisions about the future of a game’s design or development are based on evidence rather than intuition, which showcases the potential of AI—via game analytics and game data mining—for better design, development and quality assurance procedures. In summary, as we will see in the remaining chapters of this book, AI-enabled and data-driven game design can directly contribute to better games.

## 1.5 Structure of This Book

We structured this book into three main parts. In the first part (Chapter 2) we outline the core game **AI methods** that are important for the study of AI in and for games. In the second part of the book we ask the question: *how can AI be used in games?* Answers to this question define the main game AI areas identified and covered as corresponding chapters:

- AI can **play games** (Chapter 3).
- AI can **generate content** (Chapter 4).
- AI can **model players** (Chapter 5).

In the final part of the book, we attempt a synthesis of the game AI areas that make up this field and discuss the research trends in what we envisage as the **game**



**Fig. 1.2** Illustrating the associations among the chapters of this book. Light gray, blue and red areas represent chapters of Part I, Part II, and Part III of the book, respectively.

**AI panorama** (Chapter 6). Building on this synthesis, we conclude the book with a chapter dedicated to the research areas we view as largely unexplored and important for the study of game AI, namely **frontier game AI research** areas (Chapter 7). An illustration of how the different chapters fit together in this book is presented in Fig. 1.2. The readers of this book may wish to skip parts that are not of interest or not appropriate given their background. For instance, readers with a background in artificial intelligence may wish to skip the first part, while readers who wish to get a rapid overview of the game AI field or a glimpse of frontier research trends in game AI can solely focus on the final part of the book.

### 1.5.1 What We (*Don't*) Cover in This Book

The list of core uses of AI in games identified in this book should not be regarded as complete and inclusive of all potential areas of game AI research. It could also be argued that the list of areas we cover is arbitrary. However, this could likely be said of any research field in any discipline. (In software engineering, software design overlaps with software requirements analysis, and in cognitive psychology, memory research overlaps with attention research.) While it might be possible to perform an analysis of this research field so that the individual areas have minimal or no overlap, this would likely lead to a list of artificial areas that do not correspond to the areas game AI students, researchers and practitioners perceive themselves to be working in. It could also be argued that we are omitting certain areas. For example, we only briefly discuss the topic of **pathfinding** in games, whereas some other authors see

this as a core concern of game AI. In our view, pathfinding is a relatively isolated area with restricted interaction with the other uses of AI in games. Further, pathfinding has been covered substantially in other game AI textbooks [109, 461, 62] already. Another example is the area of **computational narrative**, which is viewed as a domain for content generation and covered only relatively briefly in Chapter 4. It would certainly be possible to write a whole textbook about computational narrative, but that would be someone else's job. Beyond particular application areas of game AI, in Chapter 2 we cover a number of popular methods used in the field. The list of methods is not inclusive of all AI areas that can find application in games; it is a list, however, we consider sufficient for covering the theoretical foundations of a graduate game AI course. In that regard, we only partially cover **planning** methods and **probabilistic** methods such as Bayesian approaches and methods based on Markov chains, respectively, in Chapter 2 and Chapter 4.

Another important note is the relationship of this book with the general areas of **game theory** [214, 472, 513] and other related research AI areas such as **multiagent systems** [626]. Game theory studies mathematical models of *rational* decision makers within abstract games, for the analysis of economic or social behavior in adversarial [471] or cooperative [108] settings. More specifically, game theory focuses on characterizing or predicting the actions of rational or bounded-rational agents, and studies the related emerging "game solution concepts"—such as the celebrated Nash equilibrium [478]. While the book does not cover these areas in detail, as they are peripheral to the aims of game AI as currently envisioned, we nevertheless believe that it would be fruitful to incorporate foundational ideas and concepts from game theory and multiagent systems research in the game AI field. Particular game AI areas such as game-playing and player (or opponent) modeling [214] could benefit from theoretical models of game-playing [214] and empirical implementations of agent-based systems. Similarly, we believe that game AI research and practice can only help in advancing work on theoretical game theory and multiagent systems. Of course, interweaving these fields properly with the current stream of game AI research is a non-trivial exercise, given the different focus and paths these fields have taken. Further, there are limits to the degrees theoretical models can capture the complexity of games covered in game AI. However, game theory undeniably constitutes a key theoretical pillar for the study of rational decision making, and rational decision making is arguably key for winning in games. Some instances of economic game theory that found successful applications in game AI include the various implementations of theoretical models for playing abstract, card and board games—notably a version of Poker was used as a testbed of game theory by von Neumann and Morgenstern [743] as far back as 1944 [406]. Some of these implementations are discussed in Chapter 3. Another example that bridged rational decision making theory with game AI is the *procedural personas* approach covered in Chapter 5.

Finally, note that this book is the effort of two authors with a high degree of consensus among them and not an edited volume of several authors. As such, it contains **subjective views** on how the game AI field is positioned within the greater

AI research scene and how the game AI areas are synthesized. Your mileage may vary.

## 1.6 Summary

AI has a long-standing and healthy relationship with games. AI algorithms have been advanced or even invented through games. Games, their design and development, in turn, have benefited largely by the numerous roles AI has taken in games. This book focuses on the main uses of AI in games, namely, for playing games, for generating content, and for modeling players, which are covered extensively in the following chapters. Before delving into the details of these AI uses, in the next chapter we outline the core methods and algorithms used in the field of game artificial intelligence.

## Chapter 2

# AI Methods

This chapter presents a number of basic AI methods that are commonly used in games, and which will be discussed and referred to in the remainder of this book. These are methods that are frequently covered in introductory AI courses—if you have taken such a course, it should have exposed you to at least half of the methods in this chapter. It should also have prepared you for easily understanding the other methods covered in this chapter.

As noted previously, this book assumes that the reader is already familiar with core AI methods at the level of an introductory university course in AI. Therefore, we recommend you to make sure that you are at least cursorily familiar with the methods presented in this chapter before proceeding to read the rest of the book. The algorithm descriptions in this chapter are **high-level descriptions** meant to refresh your memory if you have learned about the particular algorithm at some previous point, or to explain the general idea of the algorithm if you have never seen it before. Each section comes with pointers to the literature, either research papers or other textbooks, where you can find more details about each method.

In this chapter we divide the relevant parts of AI (for the purposes of the book) into six categories: ad-hoc authoring, tree search, evolutionary computation, supervised learning, reinforcement learning and unsupervised learning. In each section we discuss some of the main algorithms in general terms, and give suggestions for further reading. Throughout the chapter we use the game of *Ms Pac-Man* (Namco, 1982) (or Ms Pac-Man for simplicity) as an overarching testbed for all the algorithms we cover. For the sake of consistency, all the methods we cover are employed to **control** Ms Pac-Man’s behavior even though they can find a multitude of other uses in this game (e.g., generating content or analyzing player behavior). While a number of other games could have been used as our testbed in this chapter, we picked Ms Pac-Man for its popularity and its game design simplicity as well as for its high complexity when it comes to playing the game. It is important to remember that Ms Pac-Man is a **non-deterministic** variant of its ancestor *Pac-Man* (Namco, 1980) which implies that the movements of ghosts involve a degree of randomness.

In Section 2.1, we go through a quick overview of two key overarching components of all methods in this book: representation and utility. **Behavior authoring**,

covered in Section 2.2, refers to methods employing static ad-hoc representations without any form of search or learning such as finite state machines, behavior trees and utility-based AI. **Tree search**, covered in Section 2.3, refers to methods that search the space of future actions and build trees of possible action sequences, often in an adversarial setting; this includes the Minimax algorithm, and Monte Carlo tree search. Covered in Section 2.4, **evolutionary computation** refers to population-based global stochastic optimization algorithms such as genetic algorithms, or evolution strategies. **Supervised learning** (see Section 2.5) refers to learning a model that maps instances of datasets to target values such as classes; target values are necessary for supervised learning. Common algorithms used here are backpropagation (artificial neural networks), support vector machines, and decision tree learning. **Reinforcement learning** is covered in Section 2.6 and refers to methods that solve reinforcement learning problems, where a sequence of actions is associated with positive or negative rewards, but not with a “target value” (the correct action). The paradigmatic algorithm here is temporal difference (TD) learning and its popular instantiation Q-learning. Section 5.6.3 outlines **unsupervised learning** which refers to algorithms that find patterns (e.g., clusters) in datasets that do *not* have target values. This includes clustering methods such as k-means, hierarchical clustering and self-organizing maps as well as frequent pattern mining methods such as Apriori and generalized sequential patterns. The chapter concludes with a number of notable algorithms that combine elements of the algorithms above to yield **hybrid** methods. In particular we cover neuroevolution and TD learning with ANN function approximation as the most popular hybrid algorithms used in the field of game AI.

## 2.1 General Notes

Before detailing each of the algorithm types we outline two overarching elements that bind together all the AI methods covered in this book. The former is the algorithm’s **representation**; the second is its **utility**. On the one hand, any AI algorithm somehow stores and maintains knowledge obtained about a particular task at hand. On the other hand, most AI algorithms seek to find better representations of knowledge. This seeking process is driven by a utility function of some form. We should note that the utility is of no use solely in methods that employ static knowledge representations such as finite state machines or behavior trees.

### 2.1.1 Representation

Appropriately representing knowledge is a key challenge for artificial intelligence at large and it is motivated by the capacity of the human brain to store and retrieve obtained knowledge about the world. The key questions that drive the design of representations for AI are as follows. How do people represent knowledge and how

can AI potentially mimic that capacity? What is the nature of knowledge? How generic can a representation scheme be? General answers to the above questions, however, are far from trivial at this point.

As a response to the open *general* questions regarding knowledge and its representation, AI has identified numerous and very *specific* ways to store and retrieve information which is authored, obtained, or learned. The representation of knowledge about a task or a problem can be viewed as the computational mapping of the task under investigation. On that basis, the representation needs to store knowledge about the task in a format that a machine is able to process, such as a data structure.

To enable any form of artificial intelligence knowledge needs to be represented computationally and the ways this can happen are many. Representation types include **grammars** such as grammatical evolution, **graphs** such as finite state machines or probabilistic models, **trees** such as decision trees, behavior trees and genetic programming, **connectionism** such as artificial neural networks, **genetic** such as genetic algorithms and evolutionary strategies and **tabular** such as temporal difference learning and Q-learning. As we will see in the remainder of this book, all above representation types find dissimilar uses in games and can be associated with various game AI tasks.

One thing is certain for any AI algorithm that is tried on a particular task: the chosen representation has a major impact on the performance of the algorithm. Unfortunately, the type of representation to be chosen for a task follows the *no free lunch theorem* [756], suggesting that there is no single representation type which is ideal for the task at hand. As a general set of guidelines, however, the representation chosen should be as *simple* as possible. Simplicity usually comes as a delicate balance between computational effort and algorithm performance as either being over-detailed or over-simplistic will affect the performance of the algorithm. Furthermore, the representation chosen should be as *small* as possible given the complexity of the task at hand. Neither simplicity nor size are trivial decisions to make with respect to the representation. Good representations come with sufficient practical wisdom and empirical knowledge about the complexity and the qualitative features of the problem the AI is trying to solve.

### 2.1.2 Utility

Utility in game theory (and economics at large) is a measure of rational choice when playing a game. In general, it can be viewed as a function that is able to assist a search algorithm to decide which path to take. For that purpose, the utility function samples aspects of the search space and gathers information about the “goodness” of areas in the space. In a sense, a utility function is an approximation of the solution we try to find. In other words, it is a **measure of goodness** of the existing representation we search through.

Similar concepts to the utility include the **heuristic** used by computer science and AI as an *approximate* way to solve a problem faster when *exact* methods are too

slow to afford, in particular associated with the tree search paradigm. The concept of **fitness** is used similarly as a utility function that measures the degree to which a solution is good, primarily, in the area of evolutionary computation. In mathematical optimization, the **objective, loss, cost, or error** function is the utility function to be minimized (or maximized if that is the objective). In particular, in supervised learning the error function represents how well an approach maps training examples to target (desired) outputs. In the area of reinforcement learning and Markov decision processes instead, the utility is named **reward**, which is a function an agent attempts to maximize by learning to take the right action in a particular state. Finally, in the area of **unsupervised learning** utility is often provided internally and within the representation via e.g., competitive learning or self-organization.

Similarly to selecting an appropriate representation, the selection of a utility function follows the no free lunch theorem. A utility is generally difficult to design and sometimes the design task is basically impossible. The simplicity of its design pays off, but the completeness as well. The quality of a utility function largely depends on thorough empirical research and practical experience, which is gained within the domain under investigation.

### 2.1.3 Learning = Maximize Utility (Representation)

The utility function is the drive for search and essential for learning. On that basis, the utility function is the *training signal* of any machine learning algorithm as it offers a measure of goodness of the representation we have. Thereby it implicitly provides indications on what to do to further increase the current goodness of the presentation. Systems that do not require learning (such as AI methods that are based on ad-hoc designed representations; or expert-knowledge systems) do not require a utility. In supervised learning the utility is sampled from data—i.e., good input-output patterns. In reinforcement learning and evolutionary computation, instead, the training signal is provided by the environment—i.e., rewards for doing something well and punishments for doing something wrong. Finally, in unsupervised learning the training signal derives from the internal structure of the representation.

## 2.2 Ad-Hoc Behavior Authoring

In this section we discuss the first, and arguably the most popular, class of AI methods for game development. **Finite state machines**, **behavior trees** and **utility-based AI** are ad-hoc behavior authoring methods that have traditionally dominated the control of non-player characters in games. Their dominance is evident by the fact that the term *game AI* in the game development scene is still nowadays synonymous with the use of these methods.

### 2.2.1 Finite State Machines

A **Finite State Machine** (FSM) [230]—and FSM variants such as hierarchical FSMs—is the game AI method that dominated the control and decision making processes of non-player characters in games up until the mid-2000s.

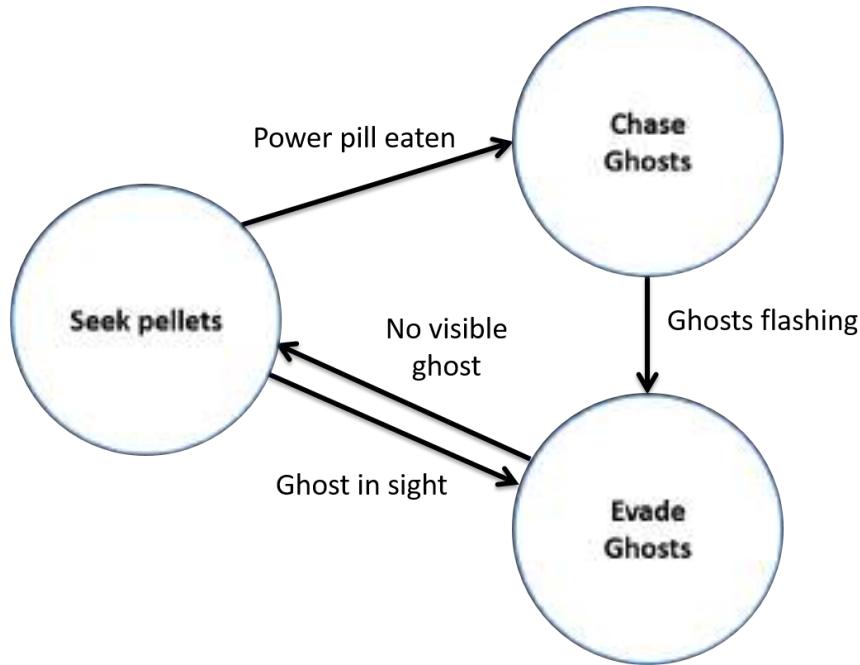
FSMs belong to the expert-knowledge systems area and are represented as graphs. An FSM graph is an abstract representation of an interconnected set of objects, symbols, events, actions or properties of the phenomenon that needs to be ad-hoc designed (represented). In particular, the graph contains nodes (states) which embed some mathematical abstraction and edges (transitions) which represent a conditional relationship between the nodes. The FSM can only be in one state at a time; the current state can change to another if the condition in the corresponding transition is fulfilled. In a nutshell, an FSM is defined by three main components:

- A number of **states** which store information about a task—e.g., you are currently on the *explore* state.
- A number of **transitions** between states which indicate a state change and are described by a condition that needs to be fulfilled—e.g., if you hear a fire shot, move to the *alerted* state.
- A set of **actions** that need to be followed within each state—e.g., while in the *explore* state *move randomly* and *seek opponents*.

FSMs are incredibly simple to design, implement, visualize, and debug. Further they have proven they work well with games over the years of their co-existence. However, they can be extremely complex to design on a large scale and are, thereby, computationally limited to certain tasks within game AI. An additional critical limitation of FSMs (and all ad-hoc authoring methods) is that they are not flexible and dynamic (unless purposely designed). After their design is completed, tested and debugged there is limited room for adaptivity and evolution. As a result, FSMs end up depicting very predictable behaviors in games. We can, in part, overcome such a drawback by representing transitions as fuzzy rules [532] or probabilities [109].

#### 2.2.1.1 An FSM for Ms Pac-Man

In this section we showcase FSMs as employed to control the Ms Pac-Man agent. A hypothetical and simplified FSM controller for Ms Pac-Man is illustrated in Fig. 2.1. In this example our FSM has three states (seek pellets, chase ghosts and evade ghosts) and four transitions (ghosts flashing, no visible ghost, ghost in sight, and power pill eaten). While in the *seek pellets* state, Ms Pac-Man moves randomly up until it detects a pellet and then follows a pathfinding algorithm to eat as many pellets as possible and as soon as possible. If a power pill is eaten, then Ms Pac-Man moves to the *chase ghosts* state in which it can use any tree-search algorithm to chase the blue ghosts. When the ghosts start flashing, Ms Pac-Man moves to the *evade ghosts* state in which it uses tree search to evade ghosts so that none is visible



**Fig. 2.1** A high-level and simplified FSM example for controlling Ms Pac-Man.

within a distance; when that happens Ms Pac-Man moves back to the *seek pellets* state.

### 2.2.2 Behavior Trees

A **Behavior Tree** (BT) [110, 112, 111] is an expert-knowledge system which, similarly to an FSM, models transitions between a finite set of tasks (or behaviors). The strength of BTs compared to FSMs is their modularity: if designed well, they can yield complex behaviors composed of simple tasks. The main difference between BT and FSMs (or even hierarchical FSMs) is that they are composed of *behaviors* rather than states. As with finite state machines, BTs are easy to design, test and debug, which made them dominant in the game development scene after their successful application in games such as *Halo 2* (Microsoft Game Studios, 2004) [291] and *Bioshock* (2K Games, 2007).

BT employs a tree structure with a root node and a number of parent and corresponding child nodes representing behaviors—see Fig. 2.2 for an example. We traverse a BT starting from the root. We then activate the execution of parent-child pairs as denoted in the tree. A child may return the following values to the parent in predetermined time steps (ticks): *run* if the behavior is still active, *success* if the

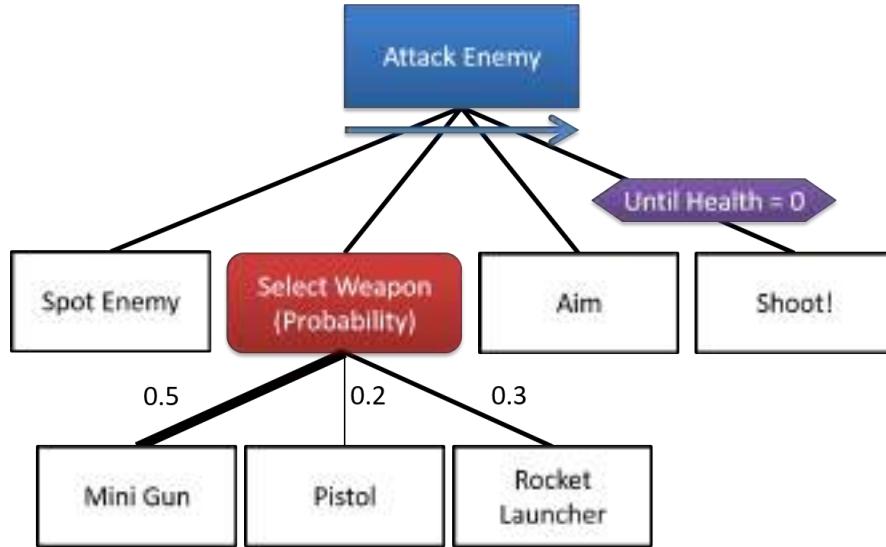
behavior is completed, *failure* if the behavior failed. BTs are composed of three node types: the **sequence**, the **selector**, and the **decorator** the basic functionality of which is described below:

- **Sequence** (see blue rectangle in Fig. 2.2): if the child behavior succeeds, the sequence continues and eventually the parent node succeeds if all child behaviors succeed; otherwise the sequence fails.
- **Selector** (see red rounded rectangle in Fig. 2.2): there are two main types of selector nodes: the *probability* and the *priority* selectors. When a probability selector is used child behaviors are selected based on parent-child probabilities set by the BT designer. On the other hand if priority selectors are used, child behaviors are ordered in a list and tried one after the other. Regardless of the selector type used, if the child behavior succeeds the selector succeeds. If the child behavior fails, the next child in the order is selected (in priority selectors) or the selector fails (in probability selectors).
- **Decorator** (see purple hexagon in Fig. 2.2): the decorator node adds complexity to and enhances the capacity of a single child behavior. Decorator examples include the number of times a child behavior runs or the time given to a child behavior to complete the task.

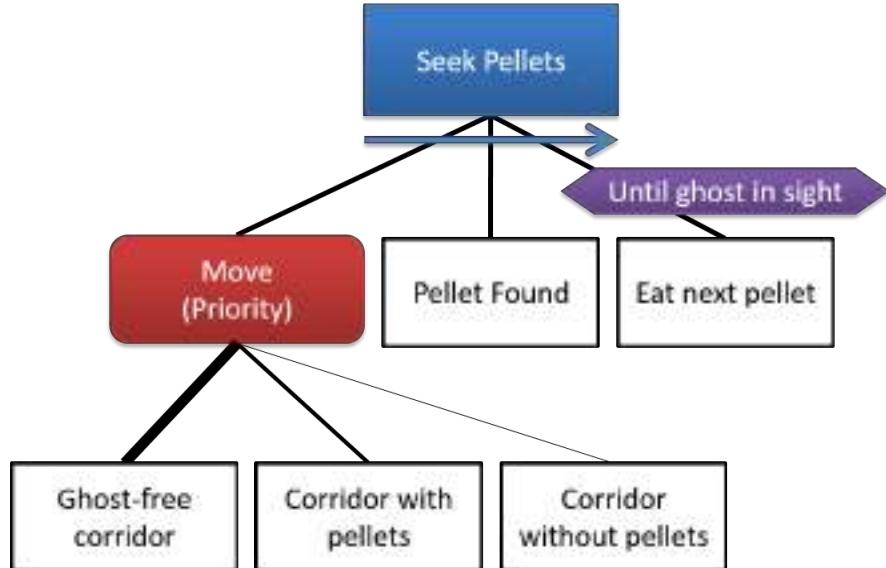
Compared to FSM, BTs are more flexible to design and easier to test; they still however suffer from similar drawbacks. In particular, their dynamicity is rather low given that they are static knowledge representations. The probability selector nodes may add to their unpredictability and methods to adapt their tree structures have already shown some promise [385]. There is also a certain degree of similarity between BTs and ABL (A Behavior Language) [440] introduced by Mateas and Stern for story-based believable characters; their dissimilarities have also been reported [749]. Note however that this section barely scratches the surface of what is possible with BT design as there are several extensions to their basic structure that help BTs improve on their modularity and their capacity to deal with more complex behavior designs [170, 627].

### 2.2.2.1 A BT for Ms Pac-Man

Similarly to the FSM example above we use Ms Pac-Man to demonstrate the use of BTs in a popular game. In Fig. 2.3 we illustrate a simple BT for the *seek pellets* behavior of Ms Pac-Man. While in the *seek pellets* sequence behavior Ms Pac-Man will first *move* (selector), it will then find a pellet and finally it will keep eating pellets until a ghost is found in sight (decorator). While in the *move* behavior—which is a priority selector—Ms Pac-Man will prioritize ghost-free corridors over corridors with pellets and over corridors without pellets.



**Fig. 2.2** A behavior tree example. The root of the BT is a sequence behavior (attack enemy) which executes the child behaviors *spot enemy*, *select weapon*, *aim* and *shoot* in sequence from left to right. The *select weapon* behavior is a probability selector giving higher probability—denoted by the thickness of the parent-child connecting lines—to the mini gun (0.5) compared to the rocket launcher (0.3) or the pistol (0.2). Once in the *shoot* behavior the decorator *until health = 0* requests the behavior to run until the enemy dies.



**Fig. 2.3** A BT example for the *seek pellets* behavior of Ms Pac-Man.

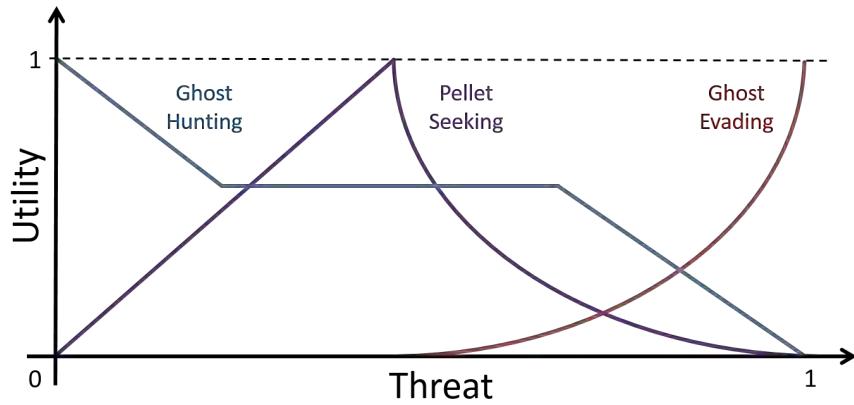
### 2.2.3 Utility-Based AI

As has been pointed out by several industrial game AI developers the lack of behavioral modularity across games and in-game tasks is detrimental for the development of high quality AI [605, 171]. An increasingly popular method for ad-hoc behavior authoring that eliminates the modularity limitations of FSMs and BTs is the **utility-based** AI approach which can be used for the design of control and decision making systems in games [425, 557]. Following this approach, instances in the game get assigned a particular utility function that gives a value for the importance of the particular instance [10, 169]. For instance, the importance of an enemy being present at a particular distance or the importance of an agent's health being low in this particular context. Given the set of all utilities available to an agent and all the options it has, utility-based AI decides which is the most important option it should consider at this moment [426]. The utility-based approach is grounded in the utility theory of economics and is based on utility function design. The approach is similar to the design of membership functions in a fuzzy set.

A utility can measure anything from observable objective data (e.g., enemy health) to subjective notions such as emotions, mood and threat. The various utilities about possible actions or decisions can be aggregated into linear or non-linear formulas and guide the agent to take decisions based on the aggregated utility. The utility values can be checked every  $n$  frames of the game. So while FSMs and BTs would examine one decision at a time, utility-based AI architectures examine all available options, assign a utility to them and select the option that is most appropriate (highest utility).

As an example of utility-based AI we will build on the one appearing in [426] for weapon selection. For selecting a weapon an agent needs to consider the following aspects: range, inertia, random noise, ammo and indoors. The *range* utility function adds value to the utility of a weapon depending on the distance—for instance, if the distance is short, pistols are assigned higher utility. *Inertia* assigns higher utility value to the current weapon so that changes of weapons are not very frequent. *Random noise* adds non-determinism to the selection so that the agent does not always pick the same weapon given the same game situation. *Ammo* returns a utility about the current level of ammunition and *indoors* penalizes the use of particular weapons indoors such as a grenade through a boolean utility function (e.g., 0 utility value if the grenade is used indoors; 1 otherwise). Our agent makes a regular check of the available weapons, assigns utility scores to all of them and selects the weapon with the best total utility.

Utility-based AI has certain advantages compared to other ad-hoc authoring techniques. It is *modular* as the decision of the game agent is dependent on a number of different factors (or considerations); this list of factors can be dynamic. Utility-based AI is also *extensible* as we can easily author new types of considerations as we see them fit. Finally, the method is *reusable* as utility components can be transferred from one decision to another and from a game to another game. As a result of these advantages utility-based AI is gradually getting traction in the game industry scene [557, 171]. Utility-based AI has seen a widespread use across game genres



**Fig. 2.4** A utility-based approach for controlling Ms Pac-Man behavior. The threat level (x-axis) is a function that lies between 0 and 1 which is based on the current position of ghosts. Ms Pac-Man considers the current level of threat, assigns utility values (through the three different curves) and decides to follow the behavior with the highest utility value. In this example the utility of *ghost evading* rises exponentially with the level of threat. The utility of *ghost hunting* decreases linearly with respect to threat up to a point where it stabilizes; it then decreases linearly as the threat level increases above a threshold value. Finally, the utility of *pellet seeking* increases linearly up to considerable threat level from which point it decreases exponentially.

and has been featured, among others, in *Kohan 2: Kings of War* (Take Two Interactive and Global Star Software, 2004), in *Iron Man* (Sega, 2008) for controlling the boss, in *Red Dead Redemption* (Rockstar Games, 2010) for weapon and dialog selection, and in *Killzone 2* (Sony Computer Entertainment, 2009) and in *F.E.A.R.* (Sierra Entertainment, 2005) for dynamic tactical decision making [426].

### 2.2.3.1 Utility-Based AI for Ms Pac-Man

Once again we use Ms Pac-Man to demonstrate the use of utility-based AI. Figure 2.4 illustrates an example of three simple utility functions that could be considered by Ms Pac-Man during play. Each function corresponds to a different behavior that is dependent on the current threat level of the game; threat is, in turn, a function of current ghost positions. At any point in the game Ms Pac-Man selects the behavior with the highest utility value.

### 2.2.3.2 A Short Note on Ad-Hoc Behavior Authoring

It is important to remember that all three methods covered in this section (and, in general, the methods covered in this chapter) represent the very **basic variants** of the algorithms. As a result, the algorithms we covered appear as static representations

of states, behaviors or utility functions. It is possible, however, to create dynamic variants of those by adding non-deterministic or fuzzy elements; for instance, one may employ fuzzy transitions in an FSM or evolve behaviors in a BT. Further, it is important to note that these ad-hoc designed architectures can feature any of the methods this book covers in the remainder of this chapter. Basic processing elements such as an FSM state, a BT behavior or a utility function or even more complex hierarchies of nodes, trees or functions can be replaced by any other AI method yielding hybrid algorithms and agent architectures. Note that possible extensions of the algorithms can be found in the work we cite in the corresponding section of each algorithm but also in the reading list we provide next.

#### 2.2.4 Further Reading

Further details on how to build and test FSMs and hierarchical FSMs can be found in [367]. For behavior trees we recommend the online tutorials and blogposts of A. Champandard found at the <http://aigamedev.com/> portal [110, 111] and recent adaptations of the basic behavior tree structure as in [627]. Finally, the book of Dave Mark [425] is a good starting point for the study of utility-based AI and its application to control and decision making in games.

When it comes to software, a BT tool has been integrated within the Unreal Engine<sup>1</sup> while several other BT Unity tools<sup>2</sup> are available for the interested reader. Further, the Behave system<sup>3</sup> streamlines the iterative process of designing, integrating and debugging behavior trees and utility-based AI.

### 2.3 Tree Search

It has been largely claimed that most, if not all, of artificial intelligence is really just search. Almost every AI problem can be cast as a search problem, which can be solved by finding the best (according to some measure) plan, path, model, function, etc. Search algorithms are therefore often seen as being at the core of AI, to the point that many textbooks (such as Russell and Norvig's famous textbook [582]) start with a treatment of search algorithms.

The algorithms presented below can all be characterized as **tree search algorithms** as they can be seen as building a **search tree** where the root is the node representing the state where the search starts. Edges in this tree represent actions the agent takes to get from one state to another, and nodes represent states. Because there are typically several different actions that can be taken in a given state, the tree

---

<sup>1</sup> <https://docs.unrealengine.com/latest/INT/Engine/>

<sup>2</sup> For instance, see <http://nodecanvas.paradoxnotion.com/> or <http://www.opsive.com/>.

<sup>3</sup> <http://eej.dk/community/documentation/behave/0-Introduction.html>

branches. Tree search algorithms mainly differ in which branches are explored and in what order.

### 2.3.1 Uninformed Search

Uninformed search algorithms are algorithms which search a state space without any further information about the goal. The basic uninformed search algorithms are commonly seen as fundamental computer science algorithms, and are sometimes not even seen as AI.

**Depth-first search** is a search algorithm which explores each branch as far as possible before backtracking and trying another branch. At every iteration of its main loop, depth-first search selects a branch and then moves on to explore the resulting node in the next iteration. When a terminal node is reached—one from which it is not possible to advance further—depth-first search advances up the list of visited nodes until it finds one which has unexplored actions. When used for playing a game, depth-first search explores the consequences of a single move until the game is won or lost, and then goes on to explore the consequences of taking a different move close to the end states.

**Breadth-first search** does the opposite of depth-first search. Instead of exploring all the consequences of a single action, breadth-first search explores all the actions from a single node before exploring any of the nodes resulting from taking those actions. So, all nodes at depth one are explored before all nodes at depth two, then all nodes at depth three, etc.

While the aforementioned are fundamental uninformed search algorithms, there are many variations and combinations of these algorithms, and new uninformed search algorithms are being developed. More information about uninformed search algorithms can be found in Chapter 4 of [582].

It is rare to see uninformed search algorithms used effectively in games, but there are exceptions such as iterative width search [58], which does surprisingly well in general video game playing, and the use of breadth-first search to evaluate aspects of strategy game maps in *Sentient Sketchbook* [379]. Also, it is often illuminating to compare the performance of state-of-the-art algorithms with a simple uninformed search algorithm.

#### 2.3.1.1 Uninformed Search for Ms Pac-Man

A depth-first approach in Ms Pac-Man would normally consider the branches of the game tree until Ms Pac-Man either completes the level or loses. The outcome of this search for each possible action would determine which action to take at a given moment. Breadth-first instead would first explore all possible actions of Ms Pac-Man at the current state of the game (e.g., going left, up, down or right) and

would then explore all their resulting nodes (children) and so on. The game tree of either method is too big and complex to visualize within a Ms Pac-Man example.

### 2.3.2 Best-First Search

In **best-first search**, the expansion of nodes in the search tree is informed by some knowledge about the goal state. In general, the node that is closest to the goal state by some criterion is expanded first. The most well-known best-first search algorithm is A\* (pronounced A star). The A\* algorithm keeps a list of “open” nodes, which are next to an explored node but which have not themselves been explored. For each open node, an estimate of its distance from the goal is made. New nodes are chosen to explore based on a lowest cost basis, where the cost is the distance from the origin node plus the estimate of the distance to the goal.

A\* can easily be understood as navigation in two- or three-dimensional space. Variants of this algorithm are therefore commonly used for **pathfinding** in games. In many games, the “AI” essentially amounts to non-player characters using A\* pathfinding to traverse scripted points. In order to cope with large, deceptive spaces numerous modifications of this basic algorithm have been proposed, including hierarchical versions of A\* [61, 661], real-time heuristic search [82], **jump point search** for uniform-cost grids [246], 3D pathfinding algorithms [68], planning algorithms for dynamic game worlds [495] that enable the animation of crowds in collision-free paths [631] and approaches for pathfinding in navigation meshes [68, 722]. The work of Steve Rabin and Nathan Sturtevant on grid-based pathfinding [551, 662] and pathfinding architectures [550] are notable examples. Sturtevant and colleagues have also been running a dedicated competition to grid-based path-planning [665] since 2012.<sup>4</sup> For the interested reader Sturtevant [663] has released a list of benchmarks for grid-based pathfinding in games<sup>5</sup> including *Dragon Age: Origins* (Electronic Arts, 2009), *StarCraft* (Blizzard Entertainment, 1998) and *Warcraft III: Reign of Chaos* (Blizzard Entertainment, 2002).

However, A\* can also be used to search in the space of game states, as opposed to simply searching physical locations. This way, best-first search can be used for **planning** rather than just navigation. The difference is in taking the changing state of the world (rather than just the changing state of a single agent) into account. Planning with A\* can be surprisingly effective, as evidenced by the winner of the 2009 Mario AI Competition—where competitors submitted agents playing *Super Mario Bros* (Nintendo, 1985)—being based on a simple A\* planner that simply tried to get to the right end of the screen at all times [717, 705] (see also Fig. 2.5).

---

<sup>4</sup> <http://movingai.com/GPPC/>

<sup>5</sup> <http://movingai.com/benchmarks/>



**Fig. 2.5** The A\* controller of the 2009 Mario AI Competition champion by R. Baumgarten [705]. The red lines illustrate possible future trajectories considered by the A\* controller of Mario, taking the dynamic nature of the game into account.

### 2.3.2.1 Best-First Search for Ms Pac-Man

Best-first search can be applicable in Pac-Man in the form of A\*. Following the paradigm of the 2009 Mario AI competition champion, Ms Pac-Man can be controlled by an A\* algorithm that searches through possible game states within a short time frame and takes a decision on where to move next (up, down, left or right). The game state can be represented in various ways: from a very direct, yet costly, representation that takes ghost and pellet coordinates into account to an indirect representation that considers the distance to the closest ghost or pellet. Regardless of the representation chosen, A\* requires the design of a cost function that will drive the search. Relevant cost functions for Ms Pac-Man would normally reward moves to areas containing pellets and penalizing areas containing ghosts.

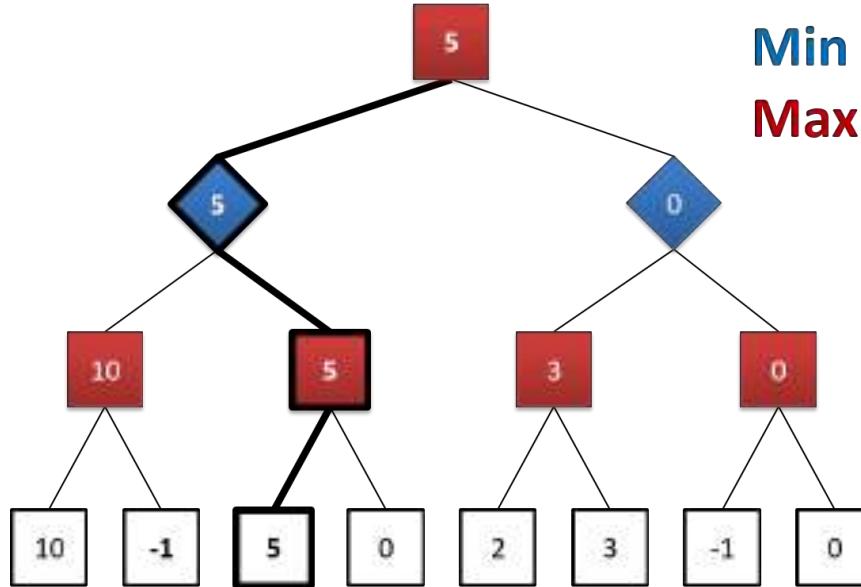
### 2.3.3 Minimax

For single-player games, simple uninformed or informed search algorithms can be used to find a path to the optimal game state. However, for two-player adversarial

games, there is another player that tries to win as well, and the actions of each player depend very much on the actions of the other player. For such games we need adversarial search, which includes the actions of two (or more) adversarial players. The basic adversarial search algorithm is called **Minimax**. This algorithm has been used very successfully for playing classic perfect-information two-player board games such as Checkers and Chess, and was in fact (re)invented specifically for the purpose of building a Chess-playing program [725].

The core loop of the Minimax algorithm alternates between player 1 and player 2—such as the white and black player in Chess—named the *min* and the *max* player. For each player, all possible moves are explored. For each of the resulting states, all possible moves by the other player are also explored, and so on until all the possible combinations of moves have been explored to the point where the game ends (e.g., with a win, a loss or a draw). The result of this process is the generation of the whole game tree from the root node down to the leaves. The outcome of the game informs the utility function which is applied onto the leaf nodes. The utility function estimates how good the current game configuration is for a player. Then, the algorithm traverses up the search tree to determine what action each player would have taken at any given state by backing-up values from leaves through the branch nodes. In doing so, it assumes that each player tries to play optimally. Thus, from the standpoint of the *max* player, it tries to maximize its score, whereas *min* tries to minimize the score of *max*; hence, the name *Minimax*. In other words, a *max* node of the tree computes the max of its child values whereas a *min* node computes the min of its child values. The optimal winning strategy is then obtained for *max* if, on *min*'s turn, a win is obtainable for *max* for *all moves* that *min* can make. The corresponding optimal strategy for *min* is when a win is possible independently of what move *max* will take. To obtain a winning strategy for *max*, for instance, we start at the root of the tree and we iteratively choose the moves leading to child nodes of highest value (on *min*'s turn the child nodes with the lowest value are selected instead). Figure 2.6 illustrates the basic steps of Minimax through a simple example.

Of course, exploring all possible moves and countermoves is infeasible for any game of interesting complexity, as the size of the search tree increases exponentially with the depth of the game or the number of moves that are simulated. Indicatively, tic-tac-toe has a game tree size of  $9! = 362,880$  states which is feasible to traverse through; however, the Chess game tree has approximately  $10^{154}$  nodes which is infeasible to search through with modern computers. Therefore, almost all actual applications of the Minimax algorithm cut off search at a given depth, and use a **state evaluation** function to evaluate the desirability of each game state at that depth. For example, in Chess a simple state evaluation function would be to merely sum the number of white pieces on the board and subtract the number of black pieces; the higher this number is, the better the situation is for the white player. (Of course, much more sophisticated board evaluation functions are commonly used.) Together with improvements to the basic Minimax algorithm such as  $\alpha\text{-}\beta$  **pruning** and the use of non-deterministic state evaluation functions, some very competent programs emerged for many classic games (e.g., IBM's Deep Blue). More information about Minimax and other adversarial search algorithms can be found in Chapter 6 of [582].



**Fig. 2.6** An abstract game tree illustrating the Minimax algorithm. In this hypothetical game of two options for each player max (represented as red squares) plays first, min (represented as blue diamonds) plays second and then max plays one last time. White squares denote terminal nodes containing a winning (positive), a losing (negative) or a draw (zero) score for the max player. Following the Minimax strategy, the scores (utility) are traversed up to the root of the game tree. The optimal play for max and min is illustrated in bold. In this simple example if both players play optimally, max wins a score of 5.

### 2.3.3.1 Minimax for Ms Pac-Man

Strictly speaking, Minimax is not applicable to Ms Pac-Man as the game is non-deterministic and, thus, the Minimax tree is formally unknown. (Of course Minimax variants with heuristic evaluation functions can be eventually applicable.) Minimax is however applicable to Ms Pac-Man's deterministic ancestor, *Pac-Man* (Namco, 1980). Again strictly speaking, *Pac-Man* is a single-player adversarial game. As such Minimax is applicable only if we assume that *Pac-Man* plays against adversaries (ghosts) who make optimal decisions. It is important to note that ghosts' movements are not represented by tree nodes; instead, they are simulated based on their assumed optimal play. Game tree nodes in *Pac-Man* may represent the game state including the position of *Pac-Man*, the ghosts, and the current pellets and power pills available. The branches of the Minimax tree are the available moves of the *Pac-Man* in each game state. The terminal nodes can, for instance, feature either a binary utility (1 if *Pac-Man* completes the level; 0 if *Pac-Man* was killed by a ghost) or the final score of the game.

### 2.3.4 Monte Carlo Tree Search

There are many games which Minimax will not play well. In particular, games with a high **branching factor** (where there are many potential actions to take at any given point in time) lead to Minimax that will only ever search a very shallow tree. Another aspect of games which frequently throws spanners in the works of Minimax is when it is hard to construct a good state evaluation function. The board game Go is a deterministic, perfect information game that is a good example of both of these phenomena. Go has a branching factor of approximately 300, whereas Chess typically has around 30 actions to choose from. The positional nature of the Go game, which is all about surrounding the adversary, makes it very hard to correctly estimate the value of a given board state. For a long time, the best Go-playing programs in the world, most of which were based on Minimax, could barely exceed the playing strength of a human beginner. In 2007, **Monte Carlo Tree Search** (MCTS) was invented and the playing strength of the best Go programs increased drastically.

Beyond complex perfect information, deterministic games such as Go, Chess and Checkers, **imperfect information** games such as Battleship, Poker, Bridge and/or **non-deterministic** games such as backgammon and monopoly cannot be solved via Minimax due to the very nature of the algorithm. In such games, MCTS not only overcomes the tree size limitation of Minimax but, given sufficient computation, it approximates the Minimax tree of the game.

So how does MCTS handle high branching factors, lack of good state evaluation functions, and lack of perfect information and determinism? To begin with, it does not search all branches of the search tree to an even depth, instead it concentrates on the more promising branches. This makes it possible to search certain branches to a considerable depth even though the branching factor is high. Further, to get around the lack of good evaluation functions, determinism and imperfect information, the standard formulation of MCTS uses **rollouts** to estimate the quality of the game state, randomly playing from a game state until the end of the game to see the expected win (or loss) outcome. The utility values obtained via the random simulations may be used efficiently to adjust the policy towards a best-first strategy (a Minimax tree approximation).

At the start of a run of the MCTS algorithm, the tree consists of a single node representing the current state of the game. The algorithm then iteratively builds a search tree by adding and evaluating new nodes representing game states. This process can be interrupted at any time, rendering MCTS an **anytime** algorithm. MCTS requires only two pieces of information to operate: the *game rules* that would, in turn, yield the available moves in the game and the *terminal state evaluation*—whether that is win, a loss, a draw, or a game score. The vanilla version of MCTS does not require a heuristic function, which is, in turn, a key advantage over Minimax.

The core loop of the MCTS algorithm can be divided into four steps: **Selection**, **Expansion** (the first two steps are also known as *tree policy*), **Simulation** and **Back-propagation**. The steps are also depicted in Fig. 2.7.

**Selection:** In this phase, it is decided which node should be expanded. The process starts at the root of the tree, and continues until a node is selected which has unexpanded children. Every time a node (action) is to be selected within the existing tree a child node  $j$  is selected to maximise the UCB1 formula:

$$\text{UCB1} = \bar{X}_j + 2C_p \sqrt{\frac{2 \ln n}{n_j}} \quad (2.1)$$

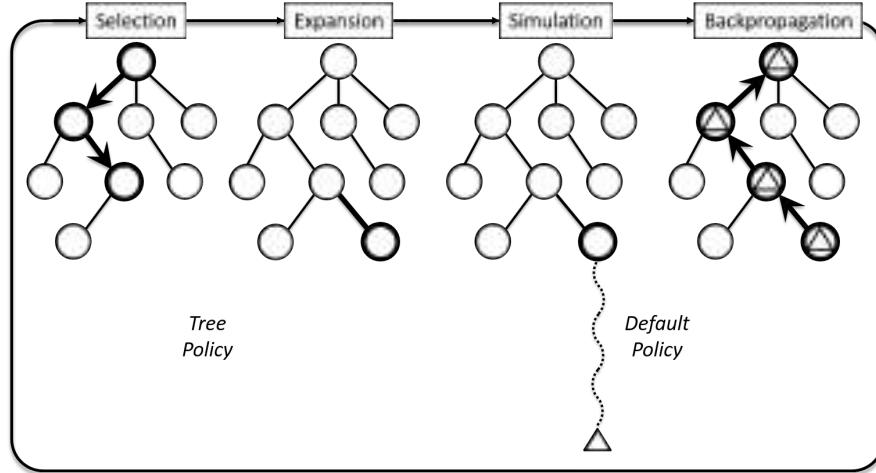
where  $\bar{X}_j$  is the average reward of all nodes beneath this node,  $C_p$  is an exploration constant (often set to  $1/\sqrt{2}$ ),  $n$  is the number of times the parent node has been visited, and  $n_j$  is the number of times the child node  $j$  has been visited. It is important to note that while UCB1 is the most popular formula used for action selection it is certainly not the only one available. Beyond equation (2.1) other options include epsilon-greedy, Thompson sampling, and Bayesian bandits. For instance, Thompson sampling selects actions stochastically based on their posterior probabilities of being optimal [692].

**Expansion:** When a node is selected that has unexpanded children—i.e., that represents a state from which actions can be taken that have not been attempted yet—one of these children is chosen for *expansion*, meaning that a simulation is done starting in that state. Selecting which child to expand is often done at random.

**Simulation (Default Policy):** After a node is expanded, a simulation (or *roll-out*) is done starting from the non-terminal node that was just expanded until the end of game to produce a value estimate. Usually, this is performed by taking random actions until a termination state is reached, i.e., until the game is either won or lost. The state at the end of the game (e.g.,  $-1$  if losing,  $+1$  if winning, but could be more nuanced) is used as the reward ( $\Delta$ ) for this simulation, and propagated up the search tree.

**Backpropagation:** The reward (the outcome of the simulation) is added to the total reward  $X$  of the new node. It is also “backed up”: added to the total reward of its parent node, its parent’s parent and so on until the root of the tree.

The simulation step might appear counter-intuitive—taking random actions seems like no good way to play a game—but it provides a relatively unbiased estimate of the quality of a game state. Essentially, the better a game state is, the more simulations are likely to end up winning the game. At least, this is true for games like Go where a game will always reach a terminal state within a certain relatively small number of moves (400 for Go). For other games like Chess, it is theoretically possible to play an arbitrary number of moves without winning or losing the game. For many video games, it is *probable* that any random sequence of actions will not end the game unless some timer runs out, meaning that most simulations will be



**Fig. 2.7** The four basic steps of MCTS exemplified through one iteration of the algorithm. The figure is a recreation of the corresponding MCTS outline figure by Chaslot et al. [118].

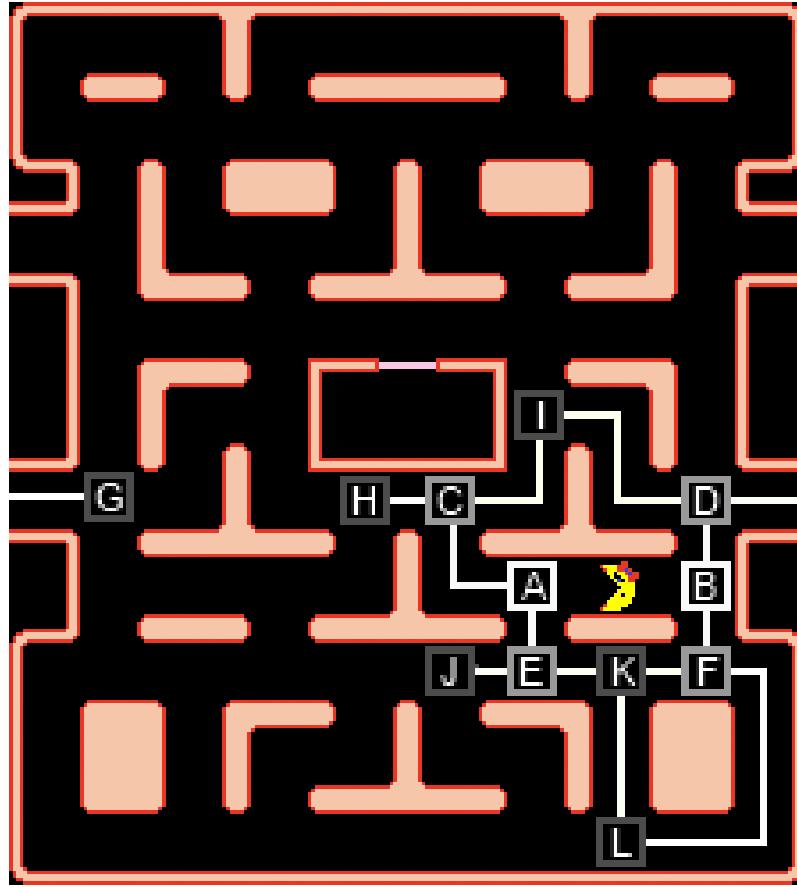
very long (tens or hundreds of thousands of steps) and not yield useful information. For example, in *Super Mario Bros* (Nintendo, 1985), the application of random actions would most likely make Mario dance around his starting point until his time is up [294]. In many cases it is therefore useful to complement the simulation step with a state evaluation function (as commonly used in Minimax), so that a simulation is performed for a set number of steps and if a terminal state is not reached a state evaluation is performed in lieu of a win-lose evaluation. In some cases it might even be beneficial to replace the simulation step entirely with a state evaluation function.

It is worth noting that there are many variations of the basic MCTS algorithm—it may in fact be more useful to see MCTS as an algorithm family or framework rather than a single algorithm.

#### 2.3.4.1 MCTS for Ms Pac-Man

MCTS can be applicable to the real-time control of the Ms Pac-Man agent. There are obviously numerous ways to represent a game state (and thereby a game tree node) and design a reward function for the game, which we will not discuss in detail here. In this section, instead, we will outline the approach followed by Pepels et al. [524] given its success in obtaining high scores for Ms Pac-Man. Their agent, named Maastricht, managed to obtain over 87,000 points and was ranked first (among 36 agents) in the Ms Pac-Man competition of the IEEE Computational Intelligence and Games conference in 2012.

When MCTS is used for real-time decision making a number of challenges become critical. First, the algorithm has limited rollout computational budget which increases the importance of heuristic knowledge. Second, the action space can be



**Fig. 2.8** The junction-based representation of a game state for the Maastricht MCTS controller [524]. All letter nodes refer to game tree nodes (decisions) for Ms Pac-Man. Imaged adapted from [524] with permission from authors.

particularly fine-grained which suggests that macro-actions are a more powerful way to model the game tree; otherwise the agent's planning will be very short-term. Third, there might be no terminal node in sight which calls for good heuristics and possibly restricting the simulation depth. The MCTS agent of Pepels et al. [524] managed to cope with all the above challenges of using MCTS for real-time control by using a restricted game tree and a junction-based game state representation (see Fig. 2.8).

### 2.3.5 Further Reading

The basic search algorithms are well covered in Russell and Norvig's classic AI textbook [582]. The A\* algorithm was invented in 1972 for robot navigation [247]; a good description of the algorithm can be found in Chapter 4 of [582]. There is plenty of more advanced material on tailoring and optimizing this algorithm for specific game problems in dedicated game AI books such as [546]. The different components of Monte Carlo tree search [141] were invented in 2006 and 2007 in the context of playing Go [142]; a good overview of and introduction to MCTS and some of its variants is given in a survey paper by Browne et al. [77].

## 2.4 Evolutionary Computation

While tree search algorithms start from the root node representing an origin state, and build a search tree based on the available actions, *optimization* algorithms do not build a search tree; they only consider complete solutions, and not the path taken to get there. As mentioned earlier in Section 2.1, all optimization algorithms assume that there is something to optimize solutions for; there must be an **objective**, alternatively called **utility function**, **evaluation function** or **fitness function**, which can assign a numerical value (the **fitness**) to a solution, which can be maximized (or minimized). Given a utility function, an optimization algorithm can be seen as an algorithm that seeks in a search space solutions that have the highest (or lowest) value of that utility.

A broad family of optimization algorithms is based on randomized variation of solutions, where one or multiple solutions are kept at any given time, and new solutions (or candidates, or search points; different terminology is used by different authors) are created through randomly changing some of the existing solutions, or maybe combining some of them. Randomized optimization algorithms which keep multiple solutions are called **evolutionary algorithms**, by analogy with natural evolution.

Another important concept when talking about optimization algorithms (and AI at large as covered in Section 2.1) is their **representation**. All solutions are represented in some way, for example, as fixed-size vectors of real numbers, or variable-length strings of characters. Generally, the same artifact can be represented in many different ways; for example, when searching for a sequence of actions that solves a maze, the action sequence can be represented in several different ways. In the most direct representation, the character at step  $t$  determines what action to take at time step  $t + 1$ . A somewhat more indirect representation for a sequence of actions would be a sequence of tuples, where the character at time step  $t$  decides what action to take and the number  $t + n$  determines for how many time steps  $n$  to take that action. The choice of representation has a big impact on the efficiency and efficacy of the search algorithm, and there are several tradeoffs at play when making these choices.

**Optimization** is an extremely general concept, and optimization algorithms are useful for a wide variety of tasks in AI as well as in computing more generally. Within AI and games, optimization algorithms such as evolutionary algorithms have been used in many roles as well. In Chapter 3 we explain how optimization algorithms can be used for searching for game-playing agents, and also for searching for action sequences (these are two very different uses of optimization that are both in the context of game-playing); in Chapter 4 we explain how we can use optimization to create game content such as levels; and in Chapter 5 we discuss how to use optimization to find player models.

### 2.4.1 Local Search

The simplest optimization algorithms are the local optimization algorithms. These are so called because they only search “locally”, in a small part of the search space, at any given time. A local optimization algorithm generally just keeps a single solution candidate at any given time, and explores variations of that solution.

The arguably simplest possible optimization algorithm is the **hill climber**. In its most common formulation, which we can call the deterministic formulation, it works as follows:

1. *Initialization:* Create a solution  $s$  by choosing a random point in search space. Evaluate its fitness.
2. Generate all possible neighbors of  $s$ . A neighbor is any solution that differs from  $s$  by at most a certain given distance (for example, a change in a single position).
3. Evaluate all the neighbors with the fitness function.
4. If none of the neighbors has a better fitness score than  $s$ , exit the algorithm and return  $s$ .
5. Otherwise, replace  $s$  with the neighbor that has the highest fitness value and go to step 2.

The deterministic hill climber is only practicable when the representation is such that each solution has a small number of neighbors. In many representations there are an astronomically high number of neighbors. It is therefore preferable to use variants of hill climbers that may guide the search effectively. One approach is the **gradient-based hill climber** that follows the gradient towards minimizing a cost function. That algorithmic approach trains artificial neural networks for instance (see Section 2.5). Another approach that we cover here is the **randomized hill climber**. This instead relies on the concept of **mutation**: a small, random change to a solution. For example, a string of letters can be mutated by randomly flipping one or two characters to some other character (see Fig. 2.9), and a vector of real



(a) **Mutation:** A number of genes is selected to be mutated with a small probability e.g., less than 1%. The selected genes are highlighted with a red outline at the top chromosome and are mutated by flipping their binary value (red genes) at the bottom chromosome.

(b) **Inversion:** Two positions in the offspring are randomly chosen and the positions between them—the gene sequence highlighted by a red outline at the top chromosome—are inverted (red genes) at the bottom chromosome.

**Fig. 2.9** Two ways of mutating a binary chromosome. In this example we use a chromosome of eleven genes. A chromosome is selected (top bit-string) and mutated (bottom bit-string).

numbers can be mutated by adding another vector to it drawn from a random distribution around zero, and with a very small standard deviation. Macro-mutations such as gene **inversion** can also be applied as visualized in Fig. 2.9. Given a representation, fitness function and mutation operator, the randomized hill climber works as follows:

1. *Initialization:* Create a solution  $s$  by choosing a random point in the search space. Evaluate its fitness.
2. *Mutation:* Generate an offspring  $s'$  by mutating  $s$ .
3. *Evaluation:* Evaluate the fitness of  $s'$ .
4. *Replacement:* If  $s'$  has higher fitness than  $s$ , replace  $s$  with  $s'$ .
5. Go to step 2.

While very simple, the randomized hill climber can be surprisingly effective. Its main limitation is that it is liable to get stuck in local optima. A **local optimum** is sort of a “dead end” in search space from which there is “no way out”; a point from which there are no better (higher-fit) points within the immediate vicinity. There are many ways of dealing with this problem. One is to simply restart the hill climber at a new randomly chosen point in the search space whenever it gets stuck. Another is **simulated annealing**, to accept moving to solutions with *lower* fitness with a given probability; this probability gradually diminishes during the search. A far more popular response to the problem of local optima is to keep not just a single solution at any time, but a **population** of solutions.

#### 2.4.1.1 Local Search for Ms Pac-Man

While we can think of a few ways one can apply local search in Ms Pac-Man we outline an example of its use for controlling path-plans. Local search could, for

instance, evolve short local plans (action sequences) of Ms Pac-Man. A solution could be represented as a set of actions that need to be taken and its fitness could be determined by the score obtained after following this sequence of actions.

### 2.4.2 Evolutionary Algorithms

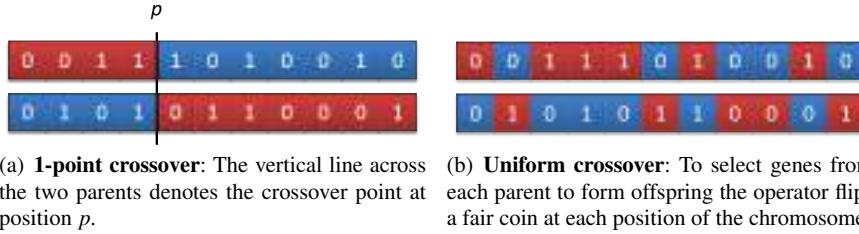
Evolutionary algorithms are randomized **global** optimization algorithms; they are called global rather than local because they search many points in the search space simultaneously, and these points can be far apart. They accomplish this by keeping a population of solutions in memory at any given time. The general idea of evolutionary computation is to optimize by “breeding” solutions: generate many solutions, throw away the bad ones and keep the good (or at least less bad) ones, and create new solutions from the good ones.

The idea of keeping a population is taken from Darwinian evolution by natural selection, from which evolutionary algorithms also get their name. The size of the population is one of the key parameters of an evolutionary algorithm; a population size of 1 yields something like a randomized hill climber, whereas populations of several thousand solutions are not unheard of.

Another idea which is taken from evolution in nature is **crossover**, also called **re-combination**. This is the equivalent of sexual reproduction in the natural world; two or more solutions (called **parents**) produce an offspring by combining elements of themselves. The idea is that if we take two good solutions, a solution that is a combination of these two—or intermediate between them—ought to be good as well, maybe even better than the parents. The offspring operator is highly dependent on the solution representation. When the solution is represented as a string or a vector, operators such as uniform crossover (which flips a fair coin and randomly picks values from each parent for each position in the offspring) or one-point crossover (where a position  $p$  in the offspring is randomly chosen, and values of positions before  $p$  are taken from parent 1 and values of positions after  $p$  are taken from parent 2) can be used. Crossover can be applied to any chromosome representation varying from a bit-string to a real-valued vector. Figure 2.10 illustrates these two crossover operators. It is in no way guaranteed, however, that the crossover operator generates an offspring that is anything as highly fit as the parents. In many cases, crossover can be highly destructive. If crossover is used, it is therefore important that the offspring operator is chosen with care for each problem. Figure 2.11 illustrates this possibility through a simple two-dimensional example.

The basic template for an evolutionary algorithm is as follows:

1. *Initialization:* The population is filled with  $N$  solutions created randomly, i.e., random points in search space. Known highly-fit solutions can also be added to this initial population.

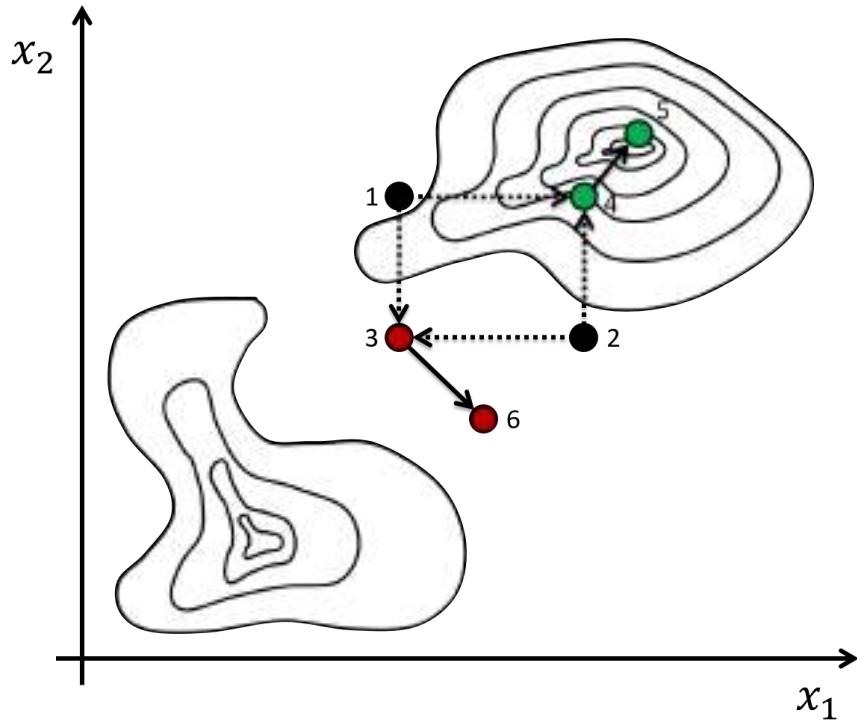


**Fig. 2.10** Two popular types of crossover used in evolutionary algorithms. In this example we use a **binary** representation and a chromosome size of eleven genes. The two bit-strings used in both crossover operators represent the two parents selected for recombination. Red and blue genes represent the two different offspring emerged from each crossover operator. Note that the operators are directly applicable to **real-valued** (floating point) representations too.

2. *Evaluation:* The fitness function is used to evaluate all solutions in the population and assign fitness values to them.
3. *Parent selection:* Based on fitness and possibly other criteria, such as distance between solutions, those population members that will be used for reproduction are selected. Selection strategies include methods directly or indirectly dependent on the fitness of the solutions, including roulette-wheel (proportionally to fitness), ranking (proportionally to rank in population) and tournament.
4. *Reproduction:* Offspring are generated through crossover from parents, or through simply copying parent solutions, or some combination of these.
5. *Variation:* Mutation is applied to some or all of the parents and/or offspring.
6. *Replacement:* In this step, we select which of the parents and/or offspring will make it to the next generation. Popular replacement strategies of the current population include the **generational** (parents die; offspring replace them), **steady state** (offspring replaces worst parent if and only if offspring is better) and **elitism** (generational, but best  $x\%$  of parents survive) approaches.
7. *Termination:* Are we done yet? Decide based on how many generations or evaluations have elapsed (**exhaustion**), the highest fitness attained by any solution (**success**), and/or some other termination condition.
8. Go to step 2.

Every iteration of the main loop (i.e., every time we reach step 2) is called a **generation**, keeping with the nature-inspired terminology. The total number of fitness evaluations performed is typically proportional to the size of the population times the number of generations.

This high-level template can be implemented and expanded in a myriad different ways; there are thousands of evolutionary or evolution-like algorithms out there, and



**Fig. 2.11** An illustration of the mutation and crossover operators in a simplified two-dimensional fitness landscape. The problem is represented by two real-valued variables ( $x_1$  and  $x_2$ ) that define the two genes of the vector chromosome. The fitness landscape is represented by the contour lines on the 2D plane. Chromosomes 1 and 2 are selected to be parents. They are recombined via 1-point crossover (dotted arrows) which yields offspring 3 and 4. Both offspring are mutated (solid arrows) to yield solutions 5 and 6. Operators that lead to poorer-fit or higher-fit solutions are, respectively, depicted with green and red color.

many of them rearrange the overall flow, add new steps and remove existing steps. In order to make this template a bit more concrete, we will give a simple example of a working evolutionary algorithm below. This is a form of **evolution strategy**, one of the main families of evolutionary algorithms. While the  $\mu + \lambda$  evolution strategy is a simple algorithm that can be implemented in 10 to 20 lines of code, it is a fully functional global optimizer and quite useful. The two main parameters are  $\mu$ , which signifies the “elite” or the size of the part of the population that is kept every generation, and  $\lambda$ , the size of the part of the population that is re-generated every generation.

1. Fill the population with  $\mu + \lambda$  randomly generated solutions.
2. Evaluate the fitness of all solutions.

3. Sort the population by decreasing fitness, so that the lowest-numbered solutions have highest fitness.
4. Remove the least fit  $\lambda$  individuals.
5. Replace the removed individuals with copies of the  $\mu$  best individuals.
6. Mutate the offspring.
7. Stop if success or exhaustion. Otherwise go to step 2.

Evolution strategies, the type of algorithms which the  $\mu + \lambda$  evolution strategy above is a simple example of, are characterized by a reliance on mutation rather than crossover to create variation, and by the use of self-adaptation to adjust mutation parameters (though that is not part of the simple algorithm above). They are also generally well suited to optimize artifacts represented as vectors of real numbers, so-called continuous optimization. Some of the very best algorithms for continuous optimization, such as the covariance matrix adaptation evolution strategy (CMA-ES) [245] and the natural evolution strategy (NES) [753], are conceptual descendants of this family of algorithms.

Another prominent family of evolutionary algorithms is **genetic algorithms** (GAs). These are characterized by a reliance on crossover rather than mutation for variation (some genetic algorithms have no mutation at all), fitness-proportional selection and solutions being often represented as bit-strings or other discrete strings. It should be noted, however, that the distinctions between different types of evolutionary algorithms are mainly based on their historical origins. These days, there are so many variations and such extensive hybridization that it often makes little sense to categorize a particular algorithm as belonging to one or the other family.

A variant of evolutionary algorithms emerges from the need of satisfying particular constraints within which a solution is not only fit but also **feasible**. When evolutionary algorithms are used for constrained optimization we are faced with a number of challenges such as that mutation and crossover cannot preserve or guarantee the feasibility of a solution. It may very well be that a mutation or a recombination between two parents may yield an infeasible offspring. One approach to deal with constraint handling is **repair**, which could be any process that turns infeasible individuals into feasible ones. A second approach is to modify the genetic operators so that the probability of an infeasible individual to appear becomes smaller. A popular approach is to merely penalize the existence of infeasible solutions by assigning them low fitness values or, alternatively, in proportion to the number of constraint violations. This strategy however may over-penalize the actual fitness of a solution which in turn will result in its rapid elimination from the population. Such a property might be undesirable and is often accused for the weak performance of evolutionary algorithms on handling constraints [456]. As a response to this limitation the **feasible-infeasible 2-population** (FI-2pop) algorithm [341] evolves two populations, one with feasible and one with infeasible solutions. The infeasible population optimizes its members towards minimizing the distance from feasibility. As the infeasible population converges to the border of feasibility, the likelihood of dis-

covering new feasible individuals increases. Feasible offspring of infeasible parents are transferred to the feasible population, boosting its diversity (and vice versa for infeasible offspring). FI-2pop has been used in games on instances where we require fit and feasible solutions such as well-designed and playable game levels [649, 379].

Finally, another blend of evolutionary algorithms considers more than one objective when attempting to find a solution to a problem. For many problems it is hard to combine all requirements and specifications into a single objective measure. It is also often true that these objectives are conflicting; for instance, if our objectives are to buy the fastest and cheapest possible laptop we will soon realize the two objectives are partially conflicting. The intuitive solution is to merely add the different objective values—as a weighted sum—and use this as your fitness under optimization. Doing so, however, has several drawbacks such as the non-trivial ad-hoc design of the weighting among the objectives, the lack of insight on the interactions between the objectives (e.g., what is the price threshold above which faster laptops are not more expensive?) and the fact that a weighted-sum single-objective approach cannot reach solutions that achieve an optimal compromise among their weighted objectives. The response to these limitations is the family of algorithms known as **multiobjective evolutionary algorithms**. A multiobjective evolutionary algorithm considers at least two objective functions—that are partially conflicting—and searches for a **Pareto front** of these objectives. The Pareto front contains solutions that cannot be improved in one objective without worsening in another. Further details about multiobjective optimization by means of evolutionary algorithms can be found in [126]. The approach is applicable in game AI on instances where more than one objective is relevant for the problem we attempt to solve: for instance, we might wish to optimize both the balance and the asymmetry of a strategy game map [712, 713], or design non-player characters that are interestingly diverse in their behavioral space [5].

#### 2.4.2.1 Evolutionary Algorithms for Ms Pac-Man

A simple way to employ evolutionary algorithms (EAs) in Ms Pac-Man is as follows. You could design a utility function based on a number of important parameters Ms Pac-Man must consider for taking the right decision on where to move next. These parameters, for instance, could be the current placement of ghosts, the presence of power pills, the number of pellets available on the level and so on. The next step would be to design a utility function as the weighted sum of these parameters. At each junction, Ms Pac-Man would need to consult its utility function for all its possible moves and pick the move with the highest utility. The weights of the utility function are unknown of course and this is where an EA can be of help by evolving the weights of the utility so that they optimize the score for Ms Pac-Man. In other words, the fitness of each chromosome (weight vector of utility) is determined by the score obtained from Ms Pac-Man within a number of simulation steps, or game levels played.

### 2.4.3 Further Reading

We recommend three books for further reading on evolutionary computation: Eiben and Smith’s *Introduction to Evolutionary Computing* [184], Ashlock’s *Evolutionary Computation for Modeling and Optimization* [21] and finally, the genetic programming field guide by Poli et al. [536].

## 2.5 Supervised Learning

Supervised learning is the algorithmic process of approximating the underlying function between labeled data and their corresponding attributes or features [49]. A popular example of supervised learning is that of a machine that is asked to distinguish between apples and pears (**labeled data**) given a set of **features** or **data attributes** such as the fruits’ color and size. Initially, the machine learns to classify between apples and pears by *seeing* a number of available fruit examples—which contain the color and size of each fruit, on one hand, and their corresponding label (apple or pear) on the other. After learning is complete, the machine should ideally be able to tell whether a new and *unseen* fruit is a pear or an apple based solely on its color and size. Beyond distinguishing between apples and pears supervised learning nowadays is used in a plethora of applications including financial services, medical diagnosis, fraud detection, web page categorization, image and speech recognition, and user modeling (among many).

Evidently, supervised learning requires a set of labeled training examples; hence supervised. More specifically, the training signal comes as a set of supervised labels on the data (e.g., this is an apple whereas that one is a pear) which acts upon a set of characterizations of these labels (e.g., this apple has red color and medium size). Consequently, each data example comes as a pair of a set of labels (or outputs) and features that correspond to these labels (or inputs). The ultimate goal of supervised learning is not to merely learn from the input-output pairs but to derive a function that approximates (better, imitates) their relationship. The derived function should be able to map well to new and *unseen* instances of input and output pairs (e.g., unseen apples and pears in our example), a property that is called **generalization**. Here are some examples of input-output pairs one can meet in games and make supervised learning relevant:  $\{\text{player health, own health, distance to player}\} \rightarrow \{\text{action (shoot, flee, idle)}\}$ ;  $\{\text{player's previous position, player's current position}\} \rightarrow \{\text{player's next position}\}$ ;  $\{\text{number of kills and headshots, ammo spent}\} \rightarrow \{\text{skill rating}\}$ ;  $\{\text{score, map explored, average heart rate}\} \rightarrow \{\text{level of player frustration}\}$ ;  $\{\text{Ms Pac-Man and ghosts position, pellets available}\} \rightarrow \{\text{Ms Pac-Man direction}\}$ .

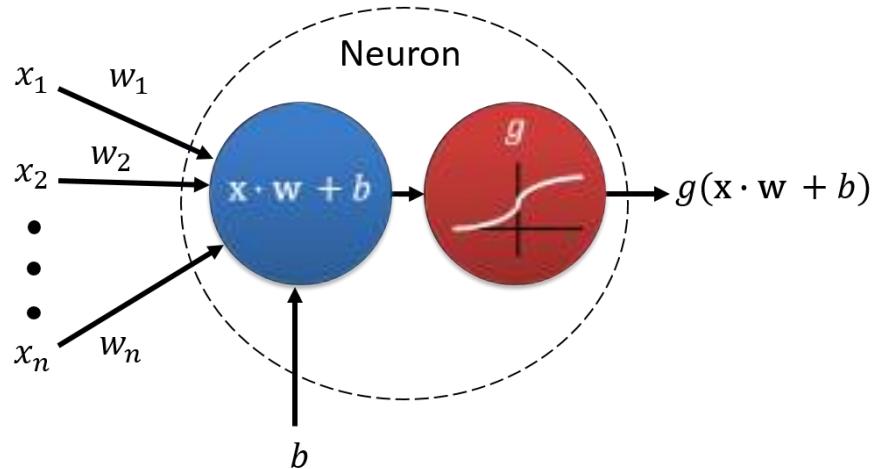
Formally, supervised learning attempts to derive a function  $f : X \rightarrow Y$ , given a set of  $N$  training examples  $\{(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_N, \mathbf{y}_N)\}$ ; where  $X$  and  $Y$  is the input and output space, respectively;  $\mathbf{x}_i$  is the feature (input) vector of the  $i$ -th example and  $\mathbf{y}_i$  is its corresponding set of labels. A supervised learning task has two core steps. In the first **training** step, the training samples—attributes and corresponding labels—

are presented and the function  $f$  between attributes and labels is derived. As we will see in the list of algorithms below  $f$  can be represented as a number of classification rules, decision trees, or mathematical formulae. In the second **testing** step  $f$  can be used to predict the labels of unknown data given their attributes. To validate the generalizability of  $f$  and to avoid overfitting to the data [49], it is common practice that  $f$  is evaluated on a new independent (test) dataset using a performance measure such as accuracy, which is the percentage of test samples that are correctly predicted by our trained function. If the accuracy is acceptable, we can use  $f$  to predict new data samples.

But how do we derive this  $f$  function? In general, an algorithmic process modifies the parameters of this function so that we achieve a good match between the given labels of our training samples and the function we attempt to approximate. There are numerous ways to find and represent that function, each one corresponding to a different supervised learning algorithm. These include artificial neural networks, case-based reasoning, decision tree learning, random forests, Gaussian regression, naive Bayes classifiers, k-nearest neighbors, and support vector machines [49]. The variety of supervised learning algorithms available is, in part, explained by the fact that there is no single learning algorithm that works best on all supervised learning problems out there. This is widely known as the *no free lunch theorem* [756].

Before covering the details of particular algorithms we should stress that the data type of the label determines the output type and, in turn, the type of the supervised learning approach that can be applied. We can identify three main types of supervised learning algorithms depending on the data type of the labels (outputs). First, we meet **classification** [49] algorithms which attempt to predict categorical class labels (discrete or nominal) such as the apples and pears of the previous example or the level in which a player will achieve her maximum score. Second, if the output data comes as an interval—such as the completion time of a game level or retention time—the supervised learning task is metric **regression** [49]. Finally, **preference learning** [215] predicts ordinal outputs such as ranks and preferences and attempts to derive the underlying global order that characterizes those ordinal labels. Examples of ordinal outputs include the ranked preferences of variant camera viewpoints, or a preference of a particular sound effect over others. The training signal in the preference learning paradigm provides information about the *relative* relation between instances of the phenomenon we attempt to approximate, whereas regression and classification provide information, respectively, about the *intensity* and the *classes* of the phenomenon.

In this book, we focus on a subset of the most promising and popular supervised learning algorithms for game AI tasks such as game playing (see Chapter 3), player behavior imitation or player preference prediction (see Chapter 5). The three algorithms outlined in the remainder of this section are artificial neural networks, support vector machines and decision tree learning. All three supervised learning algorithms covered can be used for either classification, prediction or preference learning tasks.



**Fig. 2.12** An illustration of an artificial neuron. The neuron is fed with the input vector  $\mathbf{x}$  through  $n$  connections with corresponding weight values  $\mathbf{w}$ . The neuron processes the input by calculating the weighted sum of inputs and corresponding connection weights and adding a bias weight ( $b$ ):  $\mathbf{x} \cdot \mathbf{w} + b$ . The resulting formula feeds an activation function ( $g$ ), the value of which defines the output of the neuron.

### 2.5.1 Artificial Neural Networks

**Artificial Neural Networks** (ANNs) are a bio-inspired approach for computational intelligence and machine learning. An ANN is a set of interconnected processing units (named **neurons**) which was originally designed to model the way a biological brain—containing over  $10^{11}$  neurons—processes information, operates, learns and performs in several tasks. Biological neurons have a cell body, a number of dendrites which bring information into the neuron and an axon which transmits electrochemical information outside the neuron. The artificial neuron (see Fig. 2.12) resembles the biological neuron as it has a number of **inputs**  $\mathbf{x}$  (corresponding to the neuron dendrites) each with an associated **weight** parameter  $w$  (corresponding to the synaptic strength). It also has a processing unit that combines inputs with their corresponding weights via an inner product (weighted sum) and adds a **bias** (or threshold) weight  $b$  to the weighted sum as follows:  $\mathbf{x} \cdot \mathbf{w} + b$ . This value is then fed to an **activation function**  $g$  (cell body) that yields the **output** of the neuron (corresponding to an axon terminal). ANNs are essentially simple mathematical models defining a function  $f : \mathbf{x} \rightarrow \mathbf{y}$ .

Various forms of ANNs are applicable for regression analysis, classification, and preference learning, and even unsupervised learning (via e.g., Hebbian learning [256] and self-organizing maps [347]). Core application areas include pattern recognition, robot and agent control, game-playing, decision making, gesture, speech and text recognition, medical and financial applications, affective modeling, and image recognition. The benefits of ANNs compared to other supervised learning ap-

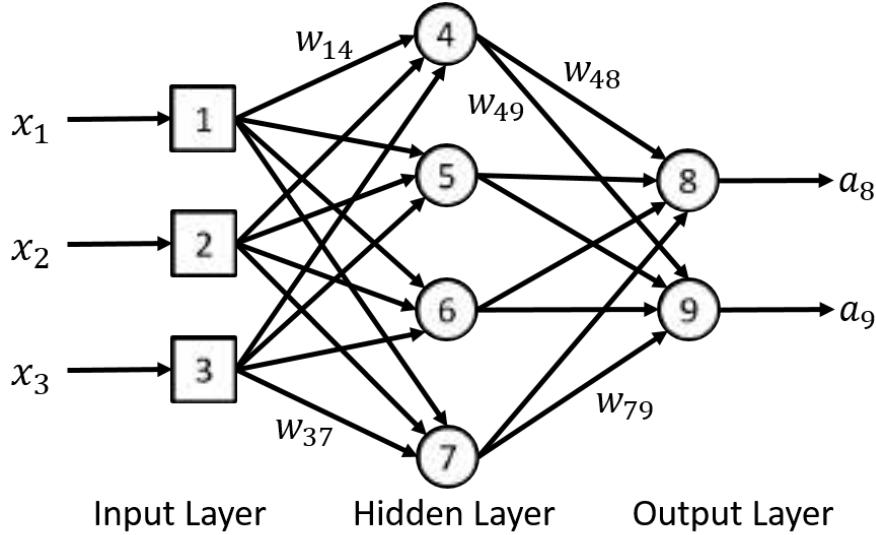
proaches is their capacity to approximate any continuous real-valued function given sufficiently large ANN architectures and computational resources [348, 152]. This capacity characterizes ANNs as *universal approximators* [279].

### 2.5.1.1 Activation Functions

Which activation function should one use in an ANN? The original model of a neuron by McCulloch and Pitts [450] in 1943 featured a Heaviside step activation function which either allows the neuron to *fire* or not. When such neurons are employed and connected to a multi-layered ANN the resulting network can merely solve linearly separable problems. The algorithm that trains such ANNs was invented in 1958 [576] and is known as the Rosenblatt's perceptron algorithm. Non-linearly separable problems such as the exclusive-or gate could only be solved after the invention of the **backpropagation** algorithm in 1975 [752]. Nowadays, there are several activation functions used in conjunction with ANNs and their training. The use of the activation function, in turn, yields different types of ANNs. Examples include Gaussian activation function that is used in radial basis function (RBF) networks [71] and the numerous types of activation functions that can be used in the compositional pattern producing networks (CPPNs) [653]. The most common function used for ANN training is the sigmoid-shaped logistic function ( $g(x) = 1/(1 + e^{-x})$ ) for the following properties: 1) it is bounded, monotonic and non-linear; 2) it is continuous and smooth and 3) its derivative is calculated trivially as  $g'(x) = g(x)(1 - g(x))$ . Given the properties above the logistic function can be used in conjunction with gradient-based optimization algorithms such as backpropagation which is described below. Other popular activation functions for training **deep** architectures of neural networks include the **rectifier**—named **rectified linear unit** (ReLU) when employed to a neuron—and its smooth approximation, the **softplus** function [231]. Compared to sigmoid-shaped activation functions, ReLUs allow for faster and (empirically) more effective training of deep ANNs, which are generally trained on large datasets (see more in Section 2.5.1.6).

### 2.5.1.2 From a Neuron to a Network

To form an ANN a number of neurons need to be structured and connected. While numerous ways have been proposed in the literature the most common of them all is to structure neurons in layers. In its simplest form, known as the **multi-layer perceptron** (MLP), neurons in an ANN are layered across one or more layers but not connected to other neurons in the same layer (see Fig. 2.13 for a typical MLP structure). The output of each neuron in each layer is connected to all the neurons in the next layer. Note that a neuron's output value feeds merely the neurons of the next layer and, thereby, becomes their input. Consequently, the outputs of the neurons in the last layer are the outputs of the ANN. The last layer of the ANN is also known as the **output layer** whereas all intermediate layers between the output



**Fig. 2.13** An MLP example with three inputs, one hidden layer containing four hidden neurons and two outputs. The ANN has labeled and ordered neurons and example connection weight labels. Bias weights  $b_j$  are not illustrated in this example but are connected to each neuron  $j$  of the ANN.

and the input are the **hidden layers**. It is important to note that the inputs of the ANN,  $\mathbf{x}$ , are connected to all the neurons of the first hidden layer. We illustrate this with an additional layer we call the **input layer**. The input layer does not contain neurons as it only distributes the inputs to the first layer of neurons. In summary, MLPs are 1) *layered* because they are grouped in layers; 2) *feed-forward* because their connections are unidirectional and always forward (from a previous layer to the next); and 3) *fully connected* because every neuron is connected to all neurons of the next layer.

#### 2.5.1.3 Forward Operation

In the previous section we defined the core components of an ANN whereas in this section we will see how we compute the output of the ANN when an input pattern is presented. The process is called **forward operation** and propagates the inputs of the ANN throughout its consecutive layers to yield the outputs. The basic steps of the forward operation are as follows:

1. Label and order neurons. We typically start numbering at the input layer and increment the numbers towards the output layer (see Fig. 2.13). Note that

- the input layer does not contain neurons, nevertheless is treated as such for numbering purposes only.
2. Label connection weights assuming that  $w_{ij}$  is the connection weight from neuron  $i$  (pre-synaptic neuron) to neuron  $j$  (post-synaptic neuron). Label bias weights that connect to neuron  $j$  as  $b_j$ .
  3. Present an input pattern  $\mathbf{x}$ .
  4. For each neuron  $j$  compute its output as follows:  $\alpha_j = g(\sum_i w_{ij} \alpha_i + b_j)$ , where  $\alpha_j$  and  $\alpha_i$  are, respectively, the output of and the inputs to neuron  $j$  (n.b.  $\alpha_i = x_i$  in the input layer);  $g$  is the activation function (usually the logistic sigmoid function).
  5. The outputs of the neurons of the output layer are the outputs of the ANN.

#### 2.5.1.4 How Does an ANN Learn?

How do we approximate  $f(\mathbf{x}; \mathbf{w}, \mathbf{b})$  so that the outputs of the ANN match the desired outputs (labels) of our dataset,  $\mathbf{y}$ ? We will need a training algorithm that adjusts the weights ( $\mathbf{w}$  and  $\mathbf{b}$ ) so that  $f : \mathbf{x} \rightarrow \mathbf{y}$ . A training algorithm as such requires two components. First, it requires a cost function to evaluate the quality of any set of weights. Second, it requires a search strategy within the space of possible solutions (i.e., the weight space). We outline these aspects in the following two subsections.

#### Cost (Error) Function

Before we attempt to adjust the weights to approximate  $f$ , we need some measure of MLP performance. The most common performance measure for training ANNs in a supervised manner is the squared Euclidean distance (error) between the vectors of the actual output of the ANN ( $\alpha$ ) and the desired labeled output  $y$  (see equation 2.2).

$$E = \frac{1}{2} \sum_j (y_j - \alpha_j)^2 \quad (2.2)$$

where the sum is taken over all the output neurons (the neurons in the final layer). Note that the  $y_j$  labels are constant values and more importantly, also note that  $E$  is a function of all the weights of the ANN since the actual outputs depend on them. As we will see below, ANN training algorithms build strongly upon this relationship between error and weights.

## Backpropagation

The **backpropagation** (or backprop) [579] algorithm is based on gradient descent optimization and is arguably the most common algorithm for training ANNs. Backpropagation stands for *backward propagation of errors* as it calculates weight updates that minimize the error function—that we defined earlier (2.2)—from the output to the input layer. In a nutshell, backpropagation computes the partial derivative (gradient) of the error function  $E$  with respect to each weight of the ANN and adjusts the weights of the ANN following the (opposite direction of the) gradient that minimizes  $E$ .

As mentioned earlier, the squared Euclidean error of (2.2) depends on the weights as the ANN output which is essentially the  $f(\mathbf{x}; \mathbf{w}, \mathbf{b})$  function. As such we can calculate the gradient of  $E$  with respect to any weight ( $\frac{\partial E}{\partial w_{ij}}$ ) and any bias weight ( $\frac{\partial E}{\partial b_j}$ ) in the ANN, which in turn will determine the degree to which the error will change if we change the weight values. We can then determine how much of such change we desire through a parameter  $\eta \in [0, 1]$  called **learning rate**. In the absence of any information about the general shape of the function between the error and the weights but the existence of information about its gradient it appears that a **gradient descent** approach would seem to be a good fit for attempting to find the global minimum of the  $E$  function. Given the lack of information about the  $E$  function, the search can start from some random point in the weight space (i.e., random initial weight values) and follow the gradient towards lower  $E$  values. This process is repeated iteratively until we reach  $E$  values we are happy with or we run out of computational resources.

More formally, the basic steps of the **backpropagation** algorithm are as follows:

1. Initialize  $\mathbf{w}$  and  $\mathbf{b}$  to random (commonly small) values.
2. For each training pattern (input-output pair):
  - (a) Present input pattern  $\mathbf{x}$ , ideally normalized to a range (e.g.,  $[0, 1]$ ).
  - (b) Compute ANN actual outputs  $\alpha_j$  using the forward operation.
  - (c) Compute  $E$  according to (2.2).
  - (d) Compute error derivatives with respect to each weight  $\frac{\partial E}{\partial w_{ij}}$  and bias weight  $\frac{\partial E}{\partial b_j}$  of the ANN from the output all the way to the input layer.
  - (e) Update weights and bias weights as  $\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}}$  and  $\Delta b_j = -\eta \frac{\partial E}{\partial b_j}$ , respectively.
3. If  $E$  is small or you are out of *computational budget*, stop! Otherwise go to step 2.

Note that we do not wish to detail the derive calculations of step 2(d) as doing so would be out of scope for this book. We instead refer the interested reader to the original backpropagation paper [579] for the exact formulas and to the reading list at the end of this section.

### Limitations and Solutions

It is worth noting that backpropagation is not guaranteed to find the global minimum of  $E$  given its local search (hill-climbing) property. Further, given its gradient-based (local) search nature, the algorithm fails to overcome potential plateaux areas in the error function landscape. As these are areas with near-zero gradient, crossing them results in near-zero weight updates and further in **premature convergence** of the algorithm. Typical solutions and enhancements of the algorithm to overcome convergence to local minima include:

- **Random restarts:** One can rerun the algorithm with new random connection weight values in the hope that the ANN is not too dependent on luck. No ANN model is good if it depends too much on luck—for instance, if it performs well only in one or two out of ten runs.
- **Dynamic learning rate:** One can either modify the learning rate parameter and observe changes in the performance of the ANN or introduce a dynamic learning rate parameter that increases when convergence is slow whereas it decreases when convergence to lower  $E$  values is fast.
- **Momentum:** Alternatively, one may add a momentum amount to the weight update rule as follows:

$$\Delta w_{ij}^{(t)} = m\Delta w_{ij}^{(t-1)} - \eta \frac{\theta E}{\theta w_{ij}} \quad (2.3)$$

where  $m \in [0, 1]$  is the momentum parameter and  $t$  is the iteration step of the weight update. The addition of a momentum value of the previous weight update ( $m\Delta w_{ij}^{(t-1)}$ ) attempts to help backpropagation to overcome a potential local minimum.

While the above solutions are directly applicable to ANNs of small size, practical wisdom and empirical evidence with modern (deep) ANN architectures, however, suggests that the above drawbacks are largely eliminated [366].

### Batch vs. Non-batch Training

Backpropagation can be employed following a batch or a non-batch learning mode. In **non-batch** mode, weights are updated every time a training sample is presented to the ANN. In **batch mode**, weights are updated after all training samples are presented to the ANN. In that case, errors are accumulated over the samples of the batch prior to the weight update. The non-batch mode is more unstable as it iteratively relies on a single data point; however, this might be beneficial for avoiding a convergence to a local minimum. The batch mode, on the other hand, is naturally a more stable gradient descent approach as weight updates are driven by the average error of all training samples in the batch. To best utilize the advantages of both approaches it is common to apply batch learning of randomly selected samples in small batch sizes.

### 2.5.1.5 Types of ANNs

Beyond the standard feedforward MLP there are numerous other types of ANN used for classification, regression, preference learning, data processing and filtering, and clustering tasks. Notably, **recurrent** neural networks (such as Hopfield networks [278], Boltzmann machines [4] and Long Short-Term Memory [266]) allow connections between neurons to form directed cycles, thus enabling an ANN to capture dynamic and temporal phenomena (e.g., time-series processing and prediction). Further, there are ANN types mostly used for clustering and data dimensionality reduction such as Kohonen self-organizing maps [347] and Autoencoders [41].

### 2.5.1.6 From Shallow to Deep

A critical parameter for ANN training is the size of the ANN. So, how *wide* and *deep* should my ANN architecture be to perform well on this particular task? While there is no formal and definite answer to this question, there is a generally accepted rule-of-thumb suggesting that the size of the network should match the complexity of the problem. According to Goodfellow et al. in their deep learning book [231] an MLP is essentially a deep (feedforward) neural network. Its *depth* is determined by the number of hidden layers it contains. Goodfellow et al. state that “It is from this terminology that the name **deep learning** arises”. On that basis, training of ANN architectures containing (at least) a hidden layer can be viewed as a deep learning task whereas single output-layered architectures can be viewed as *shallow*. Various methods have been introduced in recent years to enable training of deep architectures containing several layers. The methods largely rely on gradient search and are covered in detail in [231] for the interested reader.

### 2.5.1.7 ANNs for Ms Pac-Man

As with every other method in this chapter we will attempt to employ ANNs in the Ms Pac-Man game. One straightforward way to use ANNs in Ms Pac-Man is to attempt to imitate expert players of the game. Thus, one can ask experts to play the game and record their playthroughs, through which a number of features can be extracted and used as the input of the ANN. The resolution of the ANN input may vary from simple statistics of the game—such as the average distance between ghosts and Ms Pac-Man—to detailed pixel-to-pixel RGB values of the game level image. The output data, on the other hand, may contain the actions selected by Ms Pac-Man in each frame of the game. Given the input and desired output pairs, the ANN is trained via backpropagation to predict the action performed by expert players (ANN output) given the current game state (ANN input). The size (width and depth) of the ANN depends on both the amount of data available from the expert Ms Pac-Man players and the size of the input vector considered.

### 2.5.2 Support Vector Machines

**Support vector machines** (SVMs) [139] are an alternative and very popular set of supervised learning algorithms that can be used for classification, regression [179] and preference learning [302] tasks. A support vector machine is a binary linear classifier that is trained so as to maximize the margin between the training examples of the separate classes in the data (e.g., apples and pears). As with every other supervised learning algorithm, the attributes of new and unseen examples are seeding the SVM which predicts the class they belong to. SVMs have been used widely for text categorization, speech recognition, image classification, and hand-written character recognition among many other areas.

Similarly to ANNs, SVMs construct a hyperplane that divides the input space and represents the function  $f$  that maps between the input and the target outputs. Instead of implicitly attempting to minimize the difference between the model's actual output and the target output following the gradient of the error (as backpropagation does), SVMs construct a hyperplane that maintains the largest distance to the nearest training-data point of any other class. That distance is called a **maximum-margin** and its corresponding hyperplane divides the points ( $\mathbf{x}_i$ ) of class with label ( $y_i$ ) 1 from those with label  $-1$  in a dataset of  $n$  samples in total. In other words, the distance between the derived hyperplane and the nearest point  $\mathbf{x}_i$  from either class is maximized. Given the input attributes of a training dataset,  $\mathbf{x}$ , the general form of a hyperplane can be defined as:  $\mathbf{w} \cdot \mathbf{x} - b = 0$  where, as in backpropagation training,  $\mathbf{w}$  is the weight (normal) vector of the hyperplane and  $\frac{b}{\|\mathbf{w}\|}$  determines the offset (or weight threshold/bias) of the hyperplane from the origin (see Fig. 2.14). Thus, formally put, an SVM is a function  $f(\mathbf{x}; \mathbf{w}, b)$  that predicts target outputs ( $\mathbf{y}$ ) and attempts to

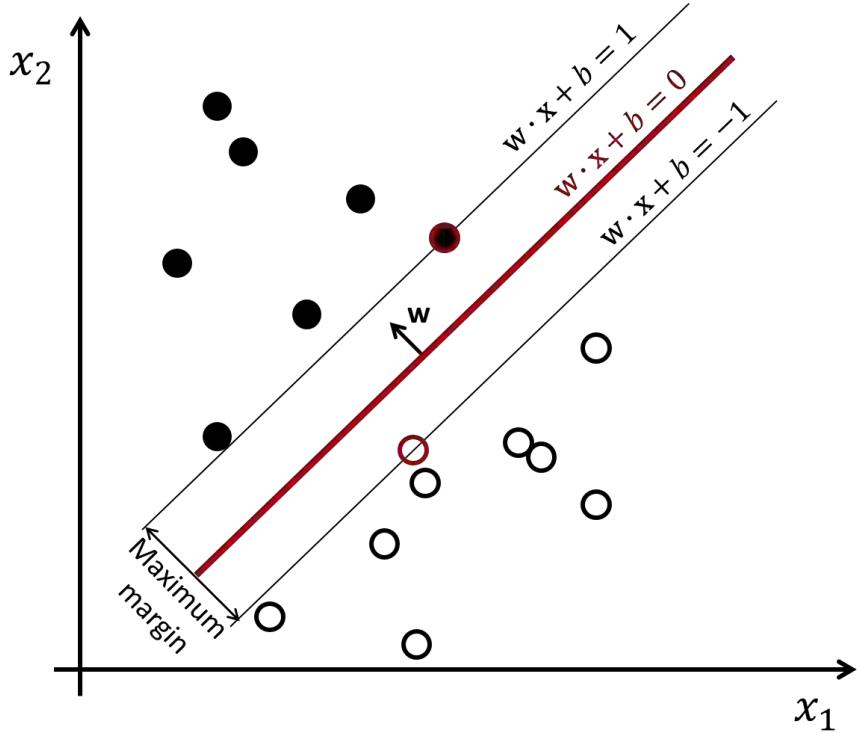
$$\text{minimize } \|\mathbf{w}\|, \quad (2.4)$$

$$\text{subject to } y_i(\mathbf{w} \cdot \mathbf{x}_i - b) \geq 1, \text{ for } i = 1, \dots, n \quad (2.5)$$

The weights  $\mathbf{w}$  and  $b$  determine the SVM classifier. The  $\mathbf{x}_i$  vectors that lie nearest to the derived hyperplane are called **support vectors**. The above problem is solvable if the training data is linearly separable (also known as a **hard-margin** classification task; see Fig. 2.14). If the data is not linearly separable (**soft-margin**) the SVM instead attempts to

$$\text{minimize } \left[ \frac{1}{n} \sum_{i=1}^n \max(0, 1 - y_i(\mathbf{w} \cdot \mathbf{x}_i - b)) \right] + \lambda \|\mathbf{w}\|^2 \quad (2.6)$$

which equals  $\lambda \|\mathbf{w}\|^2$  if the hard constraints of equation 2.5 are satisfied—i.e., if all data points are correctly classified on the right side of the margin. The value of equation (2.6) is proportional to the distance from the margin for misclassified data and  $\lambda$  is designed so as to qualitatively determine the degree to which the margin-size should be increased versus ensuring that the  $\mathbf{x}_i$  will lie on the correct side of the



**Fig. 2.14** An example of a maximum-margin hyperplane (red thick line) and margins (black lines) for an SVM which is trained on data samples from two classes. Solid and empty circles correspond to data with labels 1 and  $-1$ , respectively. The classification is mapped onto a two-dimensional input vector  $(x_1, x_2)$  in this example. The two data samples on the margin—the circles depicted with red outline—are the support vectors.

margin. Evidently, if we choose a small value for  $\lambda$  we approximate the hard-margin classifier for linearly separable data.

The standard approach for training soft-margin classifiers is to treat the learning task as a quadratic programming problem and search the space of  $w$  and  $b$  to find the widest possible margin that matches all data points. Other approaches include sub-gradient descent and coordinate descent.

In addition to linear classification tasks, SVMs can support non-linear classification by employing a number of different non-linear **kernels** which map the input space onto higher-dimensional feature spaces. The SVM task remains similar, except that every dot product is replaced by a nonlinear kernel function. This allows the algorithm to fit the maximum-margin hyperplane in a transformed feature space. Popular kernels used in conjunction with SVMs include polynomial functions, Gaussian radial basis functions or hyperbolic tangent functions.

While SVMs were originally designed to tackle **binary** classification problems there exist several SVM variants that can tackle multi-class classification [284], regression [179] and preference learning [302] that the interested reader can refer to.

SVMs have a number of advantages compared to other supervised learning approaches. They are efficient in finding solutions when dealing with large, yet sparse, datasets as they only depend on support vectors to construct hyperplanes. They also handle well large feature spaces as the learning task complexity does not depend on the dimensionality of the feature space. SVMs feature a simple convex optimization problem which can be guaranteed to converge to a single global solution. Finally, overfitting can be controlled easily through the soft margin classification approach.

### 2.5.2.1 SVMs for Ms Pac-Man

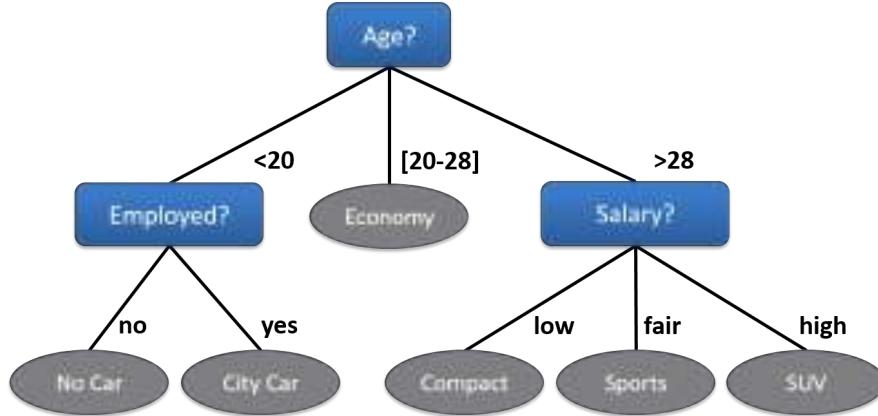
Similarly to ANNs, SVMs can be used for imitating the behavior of Ms Pac-Man expert players. The considerations about the feature (input) space and the action (output) space remain the same. In addition to the design of the input and output vectors, the size and quality of the data obtained from expert players will determine the performance of the SVM controlling Ms Pac-Man towards maximizing its score.

## 2.5.3 Decision Tree Learning

In **decision tree learning** [67], the function  $f$  we attempt to derive uses a decision tree representation which maps attributes of data observations to their target values. The former (inputs) are represented as the nodes and the latter (outputs) are represented as the leaves of the tree. The possible values of each node (input) are represented by the various branches of that node. As with the other supervised learning algorithms, decision trees can be classified depending on the output data type they attempt to learn. In particular, decision trees can be distinguished into classification, regression and rank trees if, respectively, the target output is a finite set of values, a set of continuous (interval) values, or a set of ordinal relations among observations.

An example of a decision tree is illustrated in Fig. 2.15. Tree nodes correspond to input attributes; there are branches to children for each of the possible values of each input attribute. Further leaves represent values of the output—car type in this example—given the values of the input attributes as determined by the path from the root to the leaf.

The goal of decision tree learning is to construct a mapping (a tree model) that predicts the value of target outputs based on a number of input attributes. The basic and most common approach for learning decision trees from data follows a top-down **recursive** tree induction strategy which has the characteristics of a greedy process. The algorithm assumes that both the input attributes and the target outputs have finite discrete domains and are of categorical nature. If inputs or outputs are continuous values, they can be discretized prior to constructing the tree. A tree is



**Fig. 2.15** A decision tree example: Given *age*, *employment status* and *salary* (data attributes) the tree predicts the *type of car* (target value) a person owns. Tree nodes (blue rounded rectangles) represent data attributes, or inputs, whereas leaves (gray ovals) represent target values, or outputs. Tree branches represent possible values of the corresponding parent node of the tree.

gradually constructed by splitting the available training dataset into subsets based on selections made for the attributes of the dataset. This process is repeated on a attribute-per-attribute basis in a recursive manner.

There are several variants of the above process that lead to dissimilar decision-tree algorithms. The two most notable variants of decision tree learning, however, are the **Iterative Dichotomiser 3 (ID3)** [544] and its successor **C4.5** [545]. The basic tree learning algorithm has the following general steps:

1. At start, all the training examples are at the root of the tree.
2. Select an attribute on the basis of a heuristic and pick the attribute with the maximum heuristic value. The two most popular heuristics are as follows:
  - **Information gain:** This heuristic is used by both the ID3 and the C4.5 tree-generation algorithms. Information gain  $G(A)$  is based on the concept of entropy from information theory and measures the difference in entropy  $H$  from before to after the dataset  $D$  is split on an attribute  $A$ .

$$G(A) = H(D) - H_A(D) \quad (2.7)$$

where  $H(D)$  is the entropy of  $D$  ( $H(D) = -\sum_i^m p_i \log_2(p_i)$ );  $p_i$  is the probability that an arbitrary sample in  $D$  belongs to class  $i$ ;  $m$  is the total number of classes;  $H_A(D)$  is the information needed (after using attribute  $A$  to split  $D$  into  $v$  partitions) to classify  $D$  and is calculated as  $H_A(D) = -\sum_j^v (|D_j|/|D|)H(D_j)$  with  $|x|$  being the size of  $x$ .

- **Gain ratio:** The C4.5 algorithm uses the gain ratio heuristic to reduce the bias of information gain towards attributes with a large number of values. The gain ratio normalizes information gain by taking into account the number and size of branches when choosing an attribute. The information gain ratio is the ratio between the information gain and the intrinsic value  $IV_A$  of attribute  $A$ :

$$GR(A) = G(A)/IV_A(D) \quad (2.8)$$

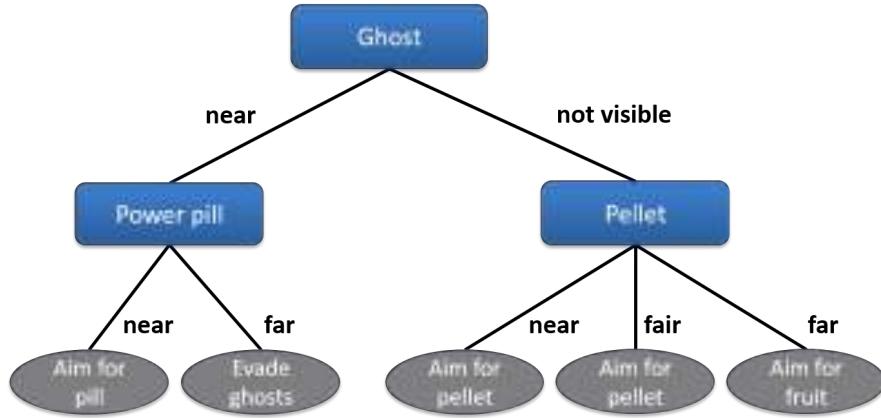
where

$$IV_A(D) = -\sum_j^v \frac{|D_j|}{|D|} \log_2\left(\frac{|D_j|}{|D|}\right) \quad (2.9)$$

3. Based on the selected attribute from step 2, construct a new node of the tree and split the dataset into subsets according to the possible values of the selected attribute. The possible values of the attribute become the branches of the node.
4. Repeat steps 2 and 3 until one of the following occurs:
  - All samples for a given node belong to the same class.
  - There are no remaining attributes for further partitioning.
  - There are no data samples left.

### 2.5.3.1 Decision Trees for Ms Pac-Man

As with ANNs and SVMs, decision tree learning requires data to be trained on. Presuming that data from expert Ms Pac-Man players would be of good quality and quantity, decision trees can be constructed to predict the strategy of Ms Pac-Man based on a number of ad-hoc designed attributes of the game state. Figure 2.16 illustrates a simplified hypothetical decision tree for controlling Ms Pac-Man. According to that example if a ghost is nearby then Ms Pac-Man checks if power pills are available in a close distance and aims for those; otherwise it takes actions so that it evades the ghost. If alternatively, ghosts are not visible Ms Pac-Man checks for pellets. If those are nearby or in a fair distance then it aims for them; otherwise it aims for the fruit, if that is available on the level. It is important to note that the leaves of the tree in our example represent control strategies (macro-actions) rather than actual actions (up, down, left, right) for Ms Pac-Man.



**Fig. 2.16** A decision tree example for controlling Ms Pac-Man. The tree is trained on data from expert Ms Pac-Man players. Given the distance from the nearest *ghost*, *power pill* and *pellet* (data attributes) the tree predicts the *strategy* Ms Pac-Man needs to follow.

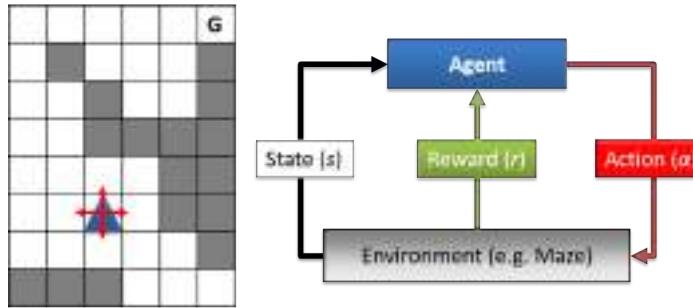
#### 2.5.4 Further Reading

The core supervised learning algorithms are covered in detail in the Russell and Norvig classic AI textbook [582] including decision tree learning (Chapter 18) and artificial neural networks (Chapter 19). Detailed descriptions of artificial neural networks and backpropagation can also be found in the book of Haykin [253]. Deep architectures of ANNs are covered in great detail in the deep learning book by Goodfellow et al. [231]. Finally, support vector machines are covered in the tutorial paper of Burges [86].

The preference learning version of backpropagation in shallow and deep architectures can be found in [430, 436] whereas RankSVM is covered in the original paper of Joachims [303].

## 2.6 Reinforcement Learning

**Reinforcement Learning** (RL) [672] is a machine learning approach inspired by behaviorist psychology and, in particular, the way humans and animals learn to take decisions via (positive or negative) rewards received by their environment. In reinforcement learning, samples of good behavior are usually not available (as in supervised learning); instead, similarly to evolutionary (reinforcement) learning, the training signal of the algorithm is provided by the environment based on how an agent is interacting with it. At a particular point in time  $t$ , the agent is on a particular **state**  $s$  and decides to take an **action**  $a$  from all the available actions in its current state. As a response the environment delivers an immediate **reward**,  $r$ . Through



**Fig. 2.17** A reinforcement learning example. The agent (triangle) attempts to reach the goal (G) by taking an action ( $a$ ) among all available actions in its current state ( $s$ ). The agent receives an immediate reward ( $r$ ) and the environment notifies the agent about its new state after taking the action.

the continuous interaction between the agent and its environment, the agent gradually learns to select actions that maximize its sum of rewards. RL has been studied from a variety of disciplinary perspectives including operations research, game theory, information theory, and genetic algorithms and has been successfully applied in problems which involve a balance between long-term and short-term rewards such as robot control and games [464, 629]. An example of the reinforcement problem is illustrated through a maze navigation task in Fig. 2.17.

More formally, the aim of the agent is to discover a **policy** ( $\pi$ ) for selecting actions that maximize a measure of a long-term reward such as the expected cumulative reward. A policy is a strategy that the agent follows in selecting actions, given the state it is in. If the function that characterizes the value of each action either exists or is learned, the optimal policy ( $\pi^*$ ) can be derived by selecting the action with the highest value. The interactions with the environment occur in discrete time steps ( $t = \{0, 1, 2, \dots\}$ ) and are modeled as a **Markov decision process** (MDP). The MDP is defined by

- $S$ : A set of states  $\{s_1, \dots, s_n\} \in S$ . The environment states are a function of the agent's information about the environment (i.e., the agent's inputs).
- $A$ : A set of actions  $\{a_1, \dots, a_m\} \in A$  possible in each state  $s$ . The actions represent the different ways the agent can act in the environment.
- $P(s, s', a)$ : The probability of transition from  $s$  to  $s'$  given  $a$ .  $P$  gives the probability of ending in state  $s'$  after picking action  $a$  in state  $s$  and it follows the **Markov property** implying that future states of the process depend only upon the present state, not on the sequence of events that preceded it. As a result, the Markov property of  $P$  makes predictions of 1-step dynamics possible.
- $R(s, s', a)$ : The reward function on transition from  $s$  to  $s'$  given  $a$ . When the agent in state  $s$  picks an action  $a$  and moves to state  $s'$ , it receives an immediate reward  $r$  from the environment.

$P$  and  $R$  define the **world model** and represent, respectively, the environment's dynamics ( $P$ ) and the long-term reward ( $R$ ) for each policy. If the world model is *known* there is no need to learn to estimate the transition probability and reward function and we thus directly calculate the optimal strategy (policy) using **model-based** approaches such as dynamic programming [44]. If, instead, the world model is *unknown* we approximate the transition and the reward functions by learning estimates of future rewards given by picking action  $a$  in state  $s$ . We then calculate our policy based on these estimates. Learning occurs via **model-free** methods such as Monte Carlo search and **temporal difference learning** [672]. In this section we put an emphasis on the latter set of algorithms and in particular, we focus on the most popular algorithm of TD learning: Q-learning. Before delving into the details of the Q-learning algorithm, we first discuss a few core RL concepts and provide a high-level taxonomy of RL algorithms according to RL problems and tools used for tackling them. We will use this taxonomy to place Q-learning with respect to RL as a whole.

### 2.6.1 Core Concepts and a High-Level Taxonomy

A central question in RL problems is the right balance between the **exploitation** of current learned knowledge versus the **exploration** of new unseen territories in the search space. Both randomly selecting actions (no exploitation) and always greedily selecting the best action according to a measure of performance or reward (no exploration) are strategies that generally yield poor results in stochastic environments. While several approaches have been proposed in the literature to address the exploration-exploitation balance issue, a popular and rather efficient mechanism for RL action selection is called  $\epsilon$ -greedy, determined by the  $\epsilon \in [0, 1]$  parameter. According to  $\epsilon$ -greedy the RL agent chooses the action it believes will return the highest future reward with probability  $1 - \epsilon$ ; otherwise, it chooses an action uniformly at random.

RL problems can be classified into **episodic** versus **incremental**. In the former class, algorithm training occurs offline and within a finite horizon of multiple training instances. The finite sequence of states, actions and reward signals received within that horizon is called an **episode**. Monte Carlo methods that rely on repeated random sampling, for instance, are a typical example of episodic RL. In the latter class of algorithms, instead, learning occurs online and it is not bounded by an horizon. We meet TD learning under incremental RL algorithms.

Another distinction is between **off-policy** and **on-policy** RL algorithms. An off-policy learner approximates the optimal policy independently of the agent's actions. As we will see below, Q-learning is an off-policy learner since it estimates the return for state-action pairs assuming that a greedy policy is followed. An on-policy RL algorithm instead approximates the policy as a process being tied to the agent's actions including the exploration steps.

**Bootstrapping** is a central notion within RL that classifies algorithms based on the way they optimize state values. Bootstrapping estimates how good a state is based on how good we think the next state is. In other words, with bootstrapping we update an estimate based on another estimate. Both TD learning and dynamic programming use bootstrapping to learn from the experience of visiting states and updating their values. Monte Carlo search methods instead do not use bootstrapping and thus learn each state value separately.

Finally, the notion of **backup** is central in RL and acts as a distinctive feature among RL algorithms. With backup we go backwards from a state in the future,  $s_{t+h}$ , to the (current) state we want to evaluate,  $s_t$ , and consider the in-between state values in our estimates. The backup operation has two main properties: its **depth**—which varies from one step backwards to a full backup—and its **breadth**—which varies from a (randomly) selected number of sample states within each time step to a full-breadth backup.

Based on the above criteria we can identify three major RL algorithm types:

1. **Dynamic programming.** In dynamic programming knowledge of the world model ( $P$  and  $R$ ) is required and the optimal policy is calculated via bootstrapping.
2. **Monte Carlo methods.** Knowledge of the world model is not required for Monte Carlo methods. Algorithms of this class (e.g., MCTS) are ideal for off-line (episodic) training and they learn via sample-breadth and full-depth backup. Monte Carlo methods do not use bootstrapping, however.
3. **TD learning.** As with Monte Carlo methods knowledge of the world model is not required and it is thus estimated. Algorithms of this type (e.g., Q-learning) learn from experience via bootstrapping and variants of backup.

In the following section we cover the most popular TD learning algorithm in the RL literature with the widest use in game AI research.

### 2.6.2 Q-Learning

**Q-learning** [748] is a model-free, off-policy, TD learning algorithm that relies on a tabular representation of  $Q(s, a)$  values (hence its name). Informally,  $Q(s, a)$  represents how good it is to pick action  $a$  in state  $s$ . Formally,  $Q(s, a)$  is the expected discounted reinforcement of taking action  $a$  in state  $s$ . The Q-learning agent learns from experience by picking actions and receiving rewards via bootstrapping.

The goal of the Q-learning agent is to maximize its expected reward by picking the right action at each state. The reward, in particular, is a weighted sum of the expected values of the discounted future rewards. The Q-learning algorithm is a simple update on the Q values in an iterative fashion. Initially, the Q table has arbitrary values as set by the designer. Then each time the agent selects an action

$a$  from state  $s$ , it visits state  $s'$ , it receives an immediate reward  $r$ , and updates its  $Q(s, a)$  value as follows:

$$Q(s, a) \leftarrow Q(s, a) + \alpha \{r + \gamma \max_{a'} Q(s', a') - Q(s, a)\} \quad (2.10)$$

where  $\alpha \in [0, 1]$  is the **learning rate** and  $\gamma \in [0, 1]$  is the **discount factor**. The learning rate determines the extent to which the new estimate for  $Q$  will override the old estimate. The discount factor weights the importance of earlier versus later rewards; the closer  $\gamma$  is to 1, the greater the weight is given to future reinforcements. As seen from equation (2.10), the algorithm uses bootstrapping since it maintains estimates of how good a state-action pair is (i.e.,  $Q(s, a)$ ) based on how good it thinks the next state is (i.e.,  $Q(s', a')$ ). It also uses a one-step-depth, full-breadth backup to estimate  $Q$  by taking into consideration all  $Q$  values of all possible actions  $a'$  of the newly visited state  $s'$ . It is proven that by using the learning rule of equation (2.10) the  $Q(s, a)$  values converge to the expected future discounted reward [748]. The optimal policy can then be calculated based on the  $Q$ -values; the agent in state  $s$  selects the action  $a$  with the highest  $Q(s, a)$  value. In summary, the basic steps of the algorithm are as follows:

Given an immediate reward function  $r$  and a table of  $Q(s, a)$  values for all possible actions in each state:

1. Initialize the table with arbitrary  $Q$  values; e.g.,  $Q(s, a) = 0$ .
2.  $s \leftarrow$  Start state.
3. While not finished\* do:
  - (a) Choose an action  $a$  based on policy derived from  $Q$  (e.g.,  $\epsilon$ -greedy).
  - (b) Apply the action, transit to state  $s'$ , and receive an immediate reward  $r$ .
  - (c) Update the value of  $Q(s, a)$  as per (2.10).
  - (d)  $s \leftarrow s'$ .

\*The most commonly used termination conditions are the algorithm's *speed*—i.e., stop within a number of iterations—or the *quality* of convergence—i.e., stop if you are satisfied with the obtained policy.

### 2.6.2.1 Limitations of Q-Learning

Q-learning has a number of limitations associated primarily with its tabular representation. First of all, depending on the chosen state-action representation the size of the state-action space might be computationally very expensive to handle. As the  $Q$  table size grows our computational needs for memory allocation and information retrieval increase. Further, we may experience very long convergence since learning time is exponential to the size of the state-action space. To overcome these

obstacles and get decent performance from RL learners we need to devise a way of reducing the state-action space. Section 2.8 outlines the approach of using artificial neural networks as Q-value function approximators, directly bypassing the Q-table limitation and yielding compressed representations for our RL learner.

### 2.6.2.2 Q-Learning for Ms Pac-Man

Q-learning is applicable for controlling Ms Pac-Man as long as we define a suitable state-action space and we design an appropriate reward function. A state in Ms Pac-Man could be represented directly as the current snapshot of the game—i.e., where Ms Pac-Man and ghosts are and which pellets and power pills are still available. That representation, however, yields a prohibitive number of game states for a Q-table to be constructed and processed. Instead, it might be preferred to choose a more indirect representation such as whether ghosts and pellets are nearby or not. Possible actions for Ms Pac-Man could be that it either keeps its current direction, it turns backward, it turns left, or it turns right. Finally, the reward function can be designed to reward Ms Pac-Man positively when it eats a pellet, a ghost or a power pill, whereas it could penalize Ms Pac-Man when it dies.

It is important to note that both Pac-Man and Ms Pac-Man follow the Markov property in the sense that any future game states may depend only upon the present game state. There is one core difference however: while the transition probability in Pac-Man is known given its deterministic nature, it is largely unknown in Ms Pac-Man given the stochastic behavior of the ghosts in that game. Thereby, Pac-Man can theoretically be solved via model-based approaches (e.g., dynamic programming) whereas the world model of Ms Pac-Man can only be approximated via model-free methods such as temporal difference learning.

### 2.6.3 Further Reading

The RL book of Sutton and Barto [672] is highly recommended for a thorough presentation of RL including Q-learning (Chapter 6). The book is freely available online.<sup>6</sup> A draft version of the latest (2017) version of the book is also available.<sup>7</sup> The survey paper of Kaelbling et al. [316] is another recommended reading of the approaches covered. Finally, for an in-depth analysis of model-based RL approaches you are referred to the dynamic programming book of Bertsekas [44].

---

<sup>6</sup> <http://incompleteideas.net/sutton/book/ebook/the-book.html>

<sup>7</sup> <http://incompleteideas.net/sutton/book/the-book-2nd.html>

## 2.7 Unsupervised Learning

As stated earlier, the utility type (or training signal) determines the class of the AI algorithm. In supervised learning the training signal is provided as data labels (target outputs) and in reinforcement learning it is derived as a reward from the environment. Unsupervised learning instead attempts to discover associations of the input by searching for patterns among all input data attributes and without having access to a target output—a machine learning process that is usually inspired by Hebbian learning [256] and the principles of self-organization [20]. With unsupervised learning we focus on the intrinsic structure of and associations in the data instead of attempting to imitate or predict target values. We cover two unsupervised learning tasks with corresponding algorithms: **clustering** and **frequent pattern mining**.

### 2.7.1 Clustering

**Clustering** is the unsupervised learning task of finding unknown groups of a number of data points so that data within a group (or else, **cluster**) is similar to each other and dissimilar to data from other clusters. Clustering has found applications in detecting groups of data across multiple attributes and in data reduction tasks such as data compression, noise smoothing, outlier detection and dataset partition. Clustering is of key importance for games with applications in player modeling, game playing and content generation.

As with classification, clustering places data into classes; the labels of the classes, however, are unknown a priori and clustering algorithms aim to discover them by assessing their quality iteratively. Since the correct clusters are unknown, similarity (and dissimilarity) depends only on the data attributes used. Good clusters are characterized by two core properties: 1) high *intra*-cluster similarity, or else, high compactness and 2) low *inter*-cluster similarity, or else, good separation. A popular measure of compactness is the average distance between every sample in the cluster and the closest representative point—e.g., centroid—as used in the k-means algorithm. Examples of separation measures include the **single link** and the **complete link**: the former is the *smallest* distance between any sample in one cluster and any sample in the other cluster; the latter is the *largest* distance between any sample in one cluster and any sample in the other cluster. While compactness and separation are objective measures of cluster validity, it is important to note that they are not indicators of cluster meaningfulness.

Beyond the validity metrics described above, clustering algorithms are defined by a **membership function** and a **search procedure**. The membership function defines the structure of the clusters in relation to the data samples. The search procedure is a strategy we follow to cluster our data given a membership function and a validity metric. Examples of such strategies include splitting all data points into clusters at once (as in k-means), or recursively merging (or splitting) clusters (as in hierarchical clustering).

Clustering can be realized via a plethora of algorithms including hierarchical clustering, k-means [411], k-medoids [329], DBSCAN [196] and self-organizing maps [347]. The algorithms are dissimilar in the way they define what a cluster is and how they form it. Selecting an appropriate clustering algorithm and its corresponding parameters, such as which distance function to use or the number of clusters to expect, depends on the aims of the study and the data available. In the remainder of the section we outline the clustering algorithms we find to be the most useful for the study of AI in games.

### 2.7.1.1 K-Means Clustering

**K-means** [411] is a vector quantization method that is considered the most popular clustering algorithm as it offers a good balance between simplicity and effectiveness. It follows a simple data partitioning approach according to which it partitions a database of objects into a set of  $k$  clusters, such that the sum of squared Euclidean distances between data points and their corresponding cluster center (centroid) is minimized—this distance is also known as the **quantization error**.

In k-means each cluster is defined by one point, that is the centroid of the cluster, and each data sample is assigned to the closest centroid. The centroid is the mean of the data samples in the cluster. The intra-cluster validity metric used by k-means is the average distance to the centroid. Initially, the data samples are randomly assigned to a cluster and then the algorithm proceeds by alternating between the re-assignment of data into clusters and the update of the resulting centroids. The basic steps of the algorithm are as follows:

Given  $k$

1. Randomly partition the data points into  $k$  nonempty clusters.
2. Compute the position of the centroids of the clusters of the current partitioning. Centroids are the centers (mean points) of the clusters.
3. Assign each data point to the cluster with the nearest centroid.
4. Stop when the assignment does not change; otherwise go to step 2.

While k-means is very popular due to its simplicity it has a number of considerable weaknesses. First, it is applicable only to data objects in a continuous space. Second, one needs to specify the number of clusters,  $k$ , in advance. Third, it is not suitable to discover clusters with non-convex shapes as it can only find hyper-spherical clusters. Finally, k-means is sensitive to outliers as data points with extremely large (or small) values may substantially distort the distribution of the data and affect the performance of the algorithm. As we will see below, hierarchical clustering manages to overcome some of the above drawbacks, suggesting a useful alternative approach to data clustering.

### 2.7.1.2 Hierarchical Clustering

Clustering methods that attempt to build a hierarchy of clusters fall under the **hierarchical clustering** approach. Generally speaking there are two main strategies available: the *agglomerative* and the *divisive*. The former constructs hierarchies in a bottom-up fashion by gradually merging data points together, whereas the latter constructs hierarchies of clusters by gradually splitting the dataset in a top-down fashion. Both clustering strategies are greedy. Hierarchical clustering uses a distance matrix as the clustering strategy (whether agglomerative or divisive). This method does not require the number of clusters  $k$  as an input, but needs a termination condition.

Indicatively, we present the basic steps of the agglomerative clustering algorithm which are as follows:

Given  $k$

1. Create one cluster per data sample.
2. Find the two closest data samples—i.e., find the shortest Euclidean distance between two points (single link)—which are not in the same cluster.
3. Merge the clusters containing these two samples.
4. Stop if there are  $k$  clusters; otherwise go to step 2.

In divisive hierarchical clustering instead, all data are initially in the same cluster which is split until every data point is on its own cluster following a split strategy—e.g., DIvisive ANAlysis Clustering (DIANA) [330]—or employing another clustering algorithm to split the data in two clusters—e.g., 2-means.

Once clusters of data are iteratively merged (or split), one can visualize the clusters by decomposing the data into several levels of nested partitioning. In other words, one can observe a tree representation of clusters which is also known as a **dendrogram**. The clustering of data is obtained by cutting the dendrogram at the desired level of squared Euclidean distance. For the interested reader, a dendrogram example is illustrated in Chapter 5.

Hierarchical clustering represents clusters as the set of data samples contained in them and, as a result, a data sample belongs to the same cluster as its closest sample. In k-means instead, each cluster is represented by a centroid and thus a data sample belongs to the cluster represented by the closest centroid. Further, when it comes to cluster validity metrics, agglomerative clustering uses the shortest distance between any sample in one cluster and a sample in another whereas k-means uses the average distance to the centroid. Due to these different algorithmic properties hierarchical clustering has the capacity to cluster data that come in any form of a connected shape; k-means, on the other hand, is only limited to hyper-spherical clusters.

### 2.7.1.3 Clustering for Ms Pac-Man

One potential application of clustering for controlling Ms Pac-Man would be to model ghost behaviors and use that information as an input to the controller of Ms Pac-Man. Whether it is k-means or hierarchical clustering, the algorithm would consider different attributes of ghost behavior—such as level exploration, behavior divergence, distance between ghosts, etc.—and cluster the ghosts into behavioral patterns or profiles. The controller of Ms Pac-Man would then consider the ghost profile met in a particular level as an additional input for guiding the agent better.

Arguably, beyond agent control, we can think of better uses of clustering for this game such as profiling Ms Pac-Man players and generating appropriate levels or challenges for them so that the game is balanced. As mentioned earlier, however, the focus of the Ms Pac-Man examples is on the control of the playing agent for the purpose of maintaining a consistent paradigm throughout this chapter.

## 2.7.2 Frequent Pattern Mining

**Frequent pattern mining** is a set of techniques that attempt to derive frequent patterns and structures in data. Patterns include sequences and itemsets. Frequent pattern mining was first proposed for mining association rules [6], which aims to identify a number of data attributes that frequently associate to each other, thereby forming conditional rules among them. There are two types of frequent pattern mining that are of particular interest for game AI: **frequent itemset mining** and **frequent sequence mining**. The former aims to find structure among data attributes that have no particular internal order whereas the latter aims to find structure among data attributes based on an inherent temporal order. While associated with the unsupervised learning paradigm, frequent pattern mining is dissimilar in both the aims and the algorithmic procedures it follows.

Popular and scalable frequent pattern mining methods include the **Apriori** algorithm [6] for itemset mining, and SPADE [793] and GSP [652, 434, 621] for sequence mining. In the remainder of this section we outline Apriori and GSP as representative algorithms for frequent itemset and frequent sequence mining, respectively.

### 2.7.2.1 Apriori

Apriori [7] is an algorithm for frequent itemset mining. The algorithm is appropriate for mining datasets that contain sets of instances (also named transactions) that each feature a set of items, or an **itemset**. Examples of transactions include books bought by an Amazon customer or apps bought by a smartphone user. The algorithm is very simple and can be described as follows: given a predetermined threshold named **support** ( $T$ ), Apriori detects the itemsets which are subsets of at least  $T$  transactions

in the database. In other words, Apriori will attempt to identify all itemsets that have at least a minimum support which is the minimum number of times an itemset exists in the dataset.

To demonstrate Apriori in a game example, below we indicatively list events from four players of an online role playing game:

- <Completed more than 10 levels; Most achievements unlocked; Bought the shield of the magi>
- <Completed more than 10 levels; Bought the shield of the magi>
- <Most achievements unlocked; Bought the shield of the magi; Found the Wizard's purple hat>
- <Most achievements unlocked; Found the Wizard's purple hat; Completed more than 10 levels; Bought the shield of the magi>

If in the example dataset above we assume that the support is 3, the following 1-itemsets (sets of only one item) can be found: <Completed more than 10 levels>, <Most achievements unlocked> and <Bought the shield of the magi>. If instead, we seek 2-itemsets with a support threshold of 3 we can find <Completed more than 10 levels, Bought the shield of the magi>, as three of the transactions above contain both of these items. Longer itemsets are not available (not frequent) for support count 3. The process can be repeated for any support threshold we wish to detect frequent itemsets for.

### 2.7.2.2 Generalized Sequential Patterns

Frequent itemset mining algorithms are not adequate if the **sequence** of events is the critical information we wish to mine from a dataset. The dataset may contain events in an ordered set of sequences such as temporal sequence data or time series. Instead, we need to opt for a frequent sequence mining approach. The sequence mining problem can be simply described as the process of finding frequently occurring subsequences given a sequence or a set of sequences.

More formally, given a dataset in which each sample is a sequence of events, namely a **data sequence**, a sequential pattern defined as a subsequence of events is a **frequent sequence** if it occurs in the samples of the dataset regularly. A frequent sequence can be defined as a sequential pattern that is supported by, at least, a minimum amount of data-sequences. This amount is determined by a threshold named minimum support value. A data sequence supports a sequential pattern if and only if it contains all the events present in the pattern in the same order. For example, the data-sequence  $\langle x_0, x_1, x_2, x_3, x_4, x_5 \rangle$  supports the pattern  $\langle x_0, x_5 \rangle$ . As with frequent itemset mining, the amount of data sequences that support a sequential pattern is referred as the **support count**.

The Generalized Sequential Patterns (GSP) algorithm [652] is a popular method for mining frequent sequences in data. GSP starts by extracting the frequent sequences with a single event, namely 1-sequences. That set of sequences is self-joined to generate all 2-sequence candidates for which we calculate their support

count. Those sequences that are frequent (i.e., their support count is greater than a threshold value) are then self-joined to generate the set of 3-sequence candidates. The algorithm is gradually increasing the length of the sequences in each algorithmic step until the next set of candidates is empty. The basic principle of the algorithm is that if a sequential pattern is frequent, then its *contiguous* subsequences are also frequent.

### 2.7.2.3 Frequent Pattern Mining for Ms Pac-Man

Patterns of events of sequences can be extracted to assist the control of Ms Pac-Man. Itemsets may be identified across successful events of expert Ms Pac-Man players given a particular support count. For instance, an Apriori algorithm running on events across several different expert players might reveal that a frequent 2-itemset is the following: <player went for the upper left corner first, player ate the bottom right power pill first>. Such information can be useful explicitly for designing rules for controlling Ms Pac-Man.

Beyond itemsets, frequencies of ghost events can be considered for playing Ms Pac-Man. For example, by running GSP on extracted attributes of ghosts it might turn out that when Ms Pac-Man eats a power pill it is very likely that the Blinky ghost moves left (<power pill, Blinky left>). Such frequent sequences can form additional inputs of any Ms Pac-Man controller—e.g., an ANN. Chapter 5 details an example on this frequent sequence mining approach in a 3D prey-predator game.

### 2.7.3 Further Reading

A general introduction to frequent pattern mining is offered in [6]. The Apriori algorithm is detailed in the original article of Agrawal and Srikant [7] whereas GSP is covered thoroughly in [652].

## 2.8 Notable Hybrid Algorithms

AI methods can be interwoven in numerous ways to yield new sophisticated algorithms that aggregate the strengths of their combined parts, often with an occurring *gestalt* effect. You can, for instance, let GAs evolve your behavior trees or FSMs; you can instead empower MCTS with ANN estimators for tree pruning; or you can add a component of local search in every search algorithm covered earlier. We name the resulting combinations of AI methods as **hybrid** algorithms and in this section we cover the two most influential, in our opinion, hybrid game AI algorithms: neuroevolution and temporal difference learning with ANN function approximators.

### 2.8.1 Neuroevolution

The evolution of artificial neural networks, or else **neuroevolution**, refers to the design of artificial neural networks—their connection weights, their topology, or both—using evolutionary algorithms [786]. Neuroevolution has been successfully applied in the domains of artificial life, robot control, generative systems and computer games. The algorithm's wide applicability is primarily due to two main reasons. First, many AI problems can be viewed as function optimization problems whose underlying general function can be approximated via an ANN. Second, neuroevolution is a method grounded in biological metaphors and evolutionary theory and inspired by the way brains evolve [567].

This evolutionary (reinforcement) learning approach is applicable either when the error function available is not differentiable or when target outputs are not available. The former may occur, for instance, when the activation functions employed in the ANN are not continuous and, thus, not differentiable. (This is a prominent phenomenon, for instance, in the compositional pattern producing networks [653].) The latter may occur in a domain for which we have no samples of good (or bad) behavior or it is impossible to define objectively what a good behavior might be. Instead of backpropagating the error and adjusting the ANN based on gradient search, neuroevolution designs ANNs via metaheuristic (evolutionary) search. In contrast to supervised learning, neuroevolution does not require a dataset of input-output pairs to train ANNs. Rather, it requires only a measure of a ANN's performance on the problem under investigation, for instance, the score of a game playing agent that is controlled by an ANN.

The core algorithmic steps of neuroevolution are as follows:

1. A population of chromosomes that represent ANNs is evolved to optimize a fitness function that characterizes the utility (quality) of the ANN representation. The population of chromosomes (ANNs) is typically initialized randomly.
2. Each chromosome is encoded into an ANN which is, in turn, tested on the task under optimization.
3. The testing procedure assigns a fitness value for each ANN of the population. The fitness of an ANN defines its measure of performance on the task.
4. Once the fitness values for all genotypes in the current population are determined, a selection strategy (e.g., roulette-wheel, tournament) is applied to pick the parents for the next generation.
5. A new population of offspring is generated by applying genetic operators on the selected ANN-encoded chromosomes. Mutation and/or crossover are applied on the chromosomes in the same way as in any evolutionary algorithm.
6. A replacement strategy (e.g., steady-state, elitism, generational) is applied to determine the final members of the new population.

7. Similarly to a typical evolutionary algorithm, the generational loop (steps 2 to 6) is repeated until we exhaust our computational budget or we are happy with the obtained fitness of the current population.

Typically there are two types of neuroevolution approaches: those that consider the evolution of a network's connection weights only and those that evolve both the connection weights and the topology of the network (including connection types and activation functions). In the former type of neuroevolution, the weight vector is encoded and represented genetically as a chromosome; in the latter type, the genetic representation includes an encoding of the ANN topology. Beyond simple MLPs, the ANN types that have been considered for evolution include the NeuroEvolution of Augmenting Topologies (NEAT) [655] and the compositional pattern producing networks [653].

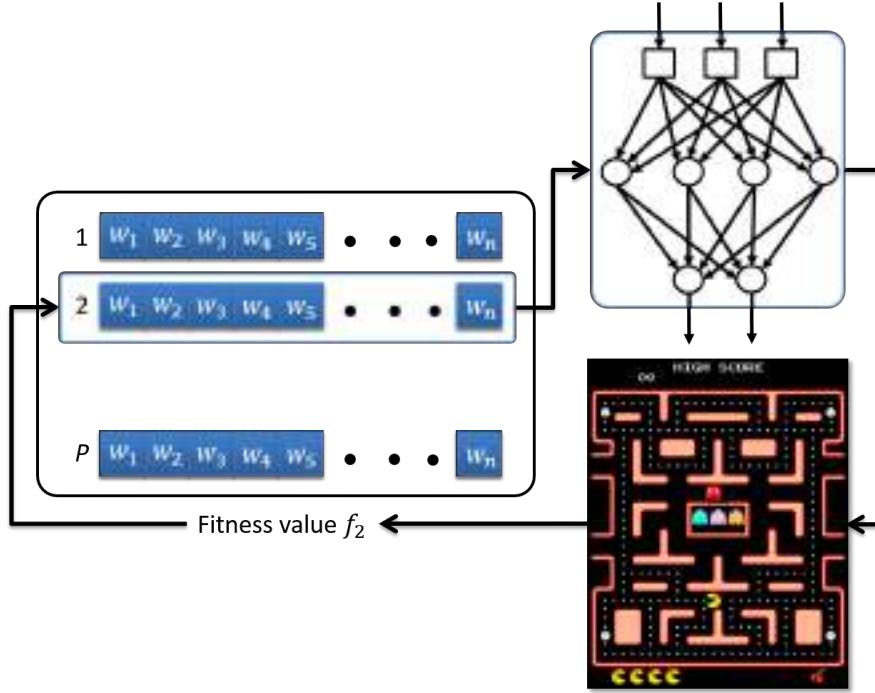
Neuroevolution has found extensive use in the games domain in roles such as those of evaluating the state-action space of a game, selecting an appropriate action, selecting among possible strategies, modeling opponent strategies, generating content, and modeling player experience [567]. The algorithm's efficiency, scalability, broad applicability, and open-ended learning are a few of the reasons that make neuroevolution a good general method for many game AI tasks [567].

### 2.8.1.1 Neuroevolution for Ms Pac-Man

One simple way to implement neuroevolution in Ms Pac-Man is to first design an ANN that considers the game state as input and output actions for Ms Pac-Man. The weights of the ANN can be evolved using a typical evolutionary algorithm and following the steps of neuroevolution as described above. The fitness of each ANN in the population is obtained by equipping Ms Pac-Man with each ANN in the population and letting her play the game for a while. The performance of the agent within that simulation time (e.g., the score) can determine the fitness value of the ANN. Figure 2.18 illustrates the steps of ANN encoding and fitness assignment in this hypothetical implementation of neuroevolution in Ms Pac-Man.

### 2.8.2 TD Learning with ANN Function Approximators

Reinforcement learning typically uses tabular representations to store knowledge. As mentioned earlier in the RL section, representing knowledge this way may drain our available computational resources since the size of the look-up table increases exponentially with respect to the action-state space. The most popular way of addressing this challenge is to use an ANN as a value (or Q value) approximator, thereby replacing the table. Doing so makes it possible to apply the algorithm to



**Fig. 2.18** Neuroevolution in Ms Pac-Man. The figure visualizes step 2 (ANN encoding) and step 3 (fitness assignment) of the algorithm for assigning a fitness value to chromosome 2 in the population (of size  $P$ ). In this example, only the weights of the ANN are evolved. The  $n$  weights of the chromosome are first encoded in the ANN and then the ANN is tested in Ms Pac-Man for a number of simulation steps (or game levels). The result of the game simulation determines the fitness value ( $f_2$ ) of the ANN.

larger spaces of action-state representations. Further, an ANN as a function approximator of  $Q$ , for instance, can handle problems with continuous state spaces which are infinitely large.

In this section, we outline two milestone examples of algorithms that utilize the ANN universal approximation capacity for temporal difference learning. The algorithms of TD-Gammon and deep Q network have been applied, respectively, to master the game of backgammon and play Atari 2600 arcade games at super-human level. Both algorithms are applicable to any RL task beyond these particular games, but the games that made them popular are used to describe the algorithms below.

### 2.8.2.1 TD-Gammon

Arguably one of the most popular success stories of AI in games is that of Tesauro's TD-Gammon software that plays backgammon on the grandmaster-level [689]. The learning algorithm was a hybrid combination of an MLP and a temporal difference

variant named TD( $\lambda$ ); see Chapter 7 of [672] for further details on the TD( $\lambda$ ) algorithm.

TD-Gammon used a standard multilayer neural network to approximate the value function. The input of the MLP was a representation of the current state of the board (Tesauro used 192 inputs) whereas the output of the MLP was the predicted probability of winning given the current state. Rewards were defined as zero for all board states except those on which the game was won. The MLP was then trained iteratively by playing the game against itself and selecting actions based on the estimated probability of winning. Each game was treated as a training episode containing a sequence of positions which were used to train the weights of the MLP by backpropagating temporal difference errors of its output.

TD-Gammon 0.0 played about 300,000 games against itself and managed to play as well as the best backgammon computer of its time. While TD-Gammon 0.0 did not win the performance horse race, it gave us a first indication of what is achievable with RL even without any backgammon expert knowledge integrated in the AI algorithm. The next iteration of the algorithm (TD-Gammon 1.0) naturally incorporated expert knowledge through specialized backgammon features that altered the input of the MLP and achieved substantially higher performance. From that point onwards the number of hidden neurons and the number of self-played games determined greatly the version of the algorithm and its resulting capacity. From TD-Gammon 2.0 (40 hidden neurons) to TD-Gammon 2.1 (80 hidden neurons) the performance of TD-Gammon gradually increased and, with TD Gammon 3.0 (160 hidden neurons), it reached the playing strength of the best human player in backgammon [689].

### 2.8.2.2 Deep Q Network

While the combination of RL and ANNs results in very powerful hybrid algorithms, the performance of the algorithm traditionally depended on the design of the input space for the ANN. As we saw earlier, even the most successful applications of RL such as the TD-Gammon agent managed to reach human-level playing performance by integrating game specific features in the input space, thereby adding expert knowledge about the game. It was up until very recently that the combination of RL and ANNs managed to reach human-level performance in a game without considering ad-hoc designed features but rather discovering them merely through learning. A team from Google’s DeepMind [464] developed a reinforcement learning agent called *deep Q network* (DQN) that trains a deep convolutional ANN via Q-learning. DQN managed to reach or exceed human-level playing performance in 29 out of 46 arcade (Atari 2600) games of the Arcade Learning Environment [40] it was trained on [464].

DQN is inspired by and based upon TD-Gammon since it uses an ANN as the function approximator for TD learning via gradient descent. As in TD-Gammon, the gradient is calculated by backpropagating the temporal difference errors. However, instead of using TD( $\lambda$ ) as the underlying RL algorithm, DQN uses Q-learning. Fur-

ther, the ANN is not a simple MLP but rather a deep convolutional neural network. DQN played each game of ALE for a large amount of frames (50 million frames). This amounts to about 38 days of playing time for each game [464].

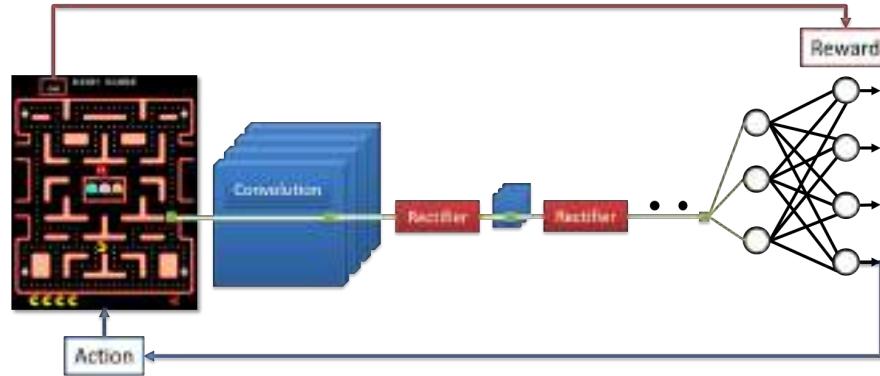
The DQN analyses a sequence of four game screens simultaneously and approximates the future game score per each possible action given its current state. In particular, the DQN uses the pixels from the four most recent game screens as its inputs, resulting in ANN input size of  $84 \times 84$  (screen size in pixels)  $\times 4$ . No other game-specific knowledge was given to the DQN beyond the screen pixel information. The architecture used for the convolutional ANN has three hidden layers that yield 32  $20 \times 20$ , 64  $9 \times 9$  and 64  $7 \times 7$  feature maps, respectively. The first (low-level) layers of the DQN process the pixels of the game screen and extract specialized visual features. The convolutional layers are followed by a fully connected hidden layer and an output layer. Each hidden layer is followed by a rectifier nonlinearity. Given a game state represented by the network's input, the outputs of the DQN are the estimated optimal action values (optimal Q-values) of the corresponding state-action pairs. The DQN is trained to approximate the Q-values (the actual score of the game) by receiving immediate rewards from the game environment. In particular, the reward is +1 if the score increases in between two successive time steps (frames), it is -1 if the score decreases, and 0 otherwise. DQN uses an  $\epsilon$ -greedy policy for its action-selection strategy. It is worth mentioning that, at the time of writing, there are newer and more efficient implementations of the deep reinforcement learning concept such as the Asynchronous Advantage Actor-Critic (A3C) algorithm [463].

### 2.8.2.3 TD Learning with ANN Function Appoximator for Ms Pac-Man

We can envisage a DQN approach for controlling Ms Pac-Man in a similar fashion to that with which ALE agents were trained [464]. A deep convolutional neural network scans the level image on a pixel-to-pixel basis (see Fig. 2.19). The image goes through a number of convolution and fully connected layers which eventually feed the input of an MLP that outputs the four possible actions for Ms Pac-Man (keep direction, move backwards, turn left, turn right). Once an action is applied, the score of the game is used as the immediate reward for updating the weights of the deep network (the convolutional ANN and the MLP). By playing for a sufficient time period the controller gathers experience (image snapshots, actions, and corresponding rewards) which trains the deep ANN to approximate a policy that maximizes the score for Ms Pac-Man.

### 2.8.3 Further Reading

For a recent thorough survey on the application of neuroevolution in games the reader may refer to [567]. For a complete review of neuroevolution please refer to Floreano et al. [205]. CPPNs and NEAT are covered in detail in [653] and [655]



**Fig. 2.19** A deep Q-learning approach for Ms Pac-Man. Following [464], the network’s first part contains a set of convolution layers which are followed by rectifier nonlinearities. The final layers of the DQN we present in this example are fully connected employing ReLUs, as in [464].

respectively. TD-Gammon and DQN are covered in detail in [689] and [464], respectively. Both are also placed within the greater RL field in the upcoming second edition of [672]. Details about the A3C algorithm can be found in [463] and implementations of the algorithm can be found directly as part of Tensorflow.

## 2.9 Summary

This chapter covered the AI methods we feel the reader of this book needs to be familiar with. We expect, however, that our readers have a basic background in AI or have completed a course in fundamentals of AI prior to reading this book. Hence, the algorithms were not covered in detail since the emphasis of this book is on the application of AI within the domain of games and not on AI per se. On that basis, we used the game of Ms Pac-Man as the overarching application testbed of all algorithms throughout this chapter.

The families of algorithms we discussed include traditional ad-hoc behavior authoring methods (such as finite state machines and behavior trees), tree search (such as best-first, Minimax and Monte Carlo tree search), evolutionary computation (such as local search and evolutionary algorithms), supervised learning (e.g., neural networks, support vector machines and decision trees), reinforcement learning (e.g., Q-learning), unsupervised learning (such as clustering and frequent pattern mining), and hybrid algorithms such as evolving artificial neural networks and artificial neural networks as approximators of expected rewards.

With this chapter we reached the end of the first, introductory, part of the book. The next part begins with a chapter on the most traditional and widely explored task of AI in games: playing!

## **Part II**

# **Ways of Using AI in Games**



## **Chapter 3**

# **Playing Games**

When most people think of AI in games they think of an AI playing the game, or controlling the non-player characters you meet in the game. This might be because of the association between AI and the idea of autonomous action, or the association between game characters and robots. While playing games is far from the only interesting application for AI in games, it is a very important one and the one with the longest history. Many methods for content generation (Chapter 4) and player modeling (Chapter 5) are also dependent on methods for playing games, and therefore it makes sense to discuss playing games before content generation and player modeling.

This chapter is devoted to AI methods for playing games, including methods for creating interesting non-player characters in games. While winning a game, appearing human-like, and providing entertainment are very different objectives, they face many of the same challenges. In fact, there are many different reasons why one might want to use AI methods to play a game. We start the chapter with discussing these various motivations (Section 3.1). Regardless of why you want to use AI to play a game, which methods you can effectively use to play the game is determined by the various characteristics of the game that, in turn, affect the choice and design of the AI method. So the next section in this chapter (Section 3.2) is devoted to characterizing games and AI algorithms according to several criteria. Once you have understood your game sufficiently, you can make an informed choice of which algorithm to play it. The following section (Section 3.3) is devoted to discussing the various methods that can be used to play games, and how the right choice of method depends on the characteristics of the game. Most of the methods discussed here will have been briefly and somewhat abstractly discussed in Chapter 2, but this chapter will go into some depth about the application of these methods to playing games.

Next, a long section (Section 3.4) divides up the space of games by game genre, and discusses how AI methods can be applied in various types of games. This section will contain plenty of examples from the literature, and some from published games. This section also introduces several commonly used game-based frameworks and competitions for testing AI game-playing algorithms. Throughout the

chapter we will mostly discuss the use of AI methods to play to **win**, but also make numerous references to the **experience**-creation aspect of game-playing.

### 3.1 Why Use AI to Play Games?

The question of why you might want to deploy some kind of artificial intelligence to play a game can be reduced to two more specific questions:

*Is the AI playing to win?*

The question here is whether achieving as high a performance as possible in the game is the overarching goal of the AI method. High performance here means getting a high score, winning over the opponent, surviving for a long time or similar. It is not always possible to define what high performance and “playing to win” means—for example, *The Sims* (Electronic Arts, 2000) has no clear winning state and the winning condition in *Minecraft* (Mojang, 2011) is not strongly related to playing the game well—but in a very large number of games, from *Tetris* (Alexey Pajitnov and Vladimir Pokhilko, 1984) to Go to the *Halo* (Microsoft Studios, 2001–2015) series, it is straightforward to define what playing better means. However, not all players play to win, and few players play to win in every game all the time. Players play to pass time, relax, test new strategies, explore the game, role-play, keep their friends company and so on (see a more detailed discussion on this topic in Chapter 5). An AI algorithm might likewise be used in a number of roles beyond simply playing as well as possible. For example, the agent might play in a human-like manner, play in an entertaining manner, or behave predictably. It is important to note that optimizing an agent for playing a game to win might be at odds with some of the other ways of playing: many high-performing AI agents play in distinctly non-human, boring and/or unpredictable ways, as we will see in some case studies.

*Is the AI taking the role of a human player?*

Some games are single-player, and some games are multi-player where all players are human. This is particularly true for classic board games. But many, probably most, video games include various non-player characters. These are controlled by the computer software in some way—in fact, for many game developers “game AI” refers to the program code that controls the NPCs, regardless of how simple or sophisticated that code is. Obviously, the role of NPCs varies sharply between games, and within games. In the discussion of this chapter we refer to non-player roles as those that a human could not take, or would not want to take. Thus, all roles in an exclusively multi-player first-person shooter (FPS) such as *Counter-Strike* (Valve Corporation, 2000) are player roles, whereas a typical single-player role-playing

|            | Player   | Non-Player   |
|------------|--|--|
| Win        | <b>Motivation</b><br>Games as AI testbeds, AI that challenges players, Simulation-based testing                    | <b>Motivation</b><br>Playing roles that humans would not (want to) play, Game balancing                            |
| Experience | <b>Motivation</b><br>Simulation-based testing, Game demonstrations   | <b>Motivation</b><br>Believable and human-like agents  |
|            | <b>Examples</b><br>Board Game AI (TD-Gammon, Chinook, Deep Blue, AlphaGo, Libratus), Jeopardy! (Watson), StarCraft | <b>Examples</b><br>Rubber banding  |
|            | <b>Examples</b><br>Game Turing Tests (2kBot, Prize/Mario), Persona Modelling                                       | <b>Examples</b><br>AI that: acts as an adversary, provides assistance, is emotively expressive, tells a story, ... |

**Fig. 3.1** Why use AI to play games? The two possible goals (win, experience) AI can aim for and the two roles (player, non-player) AI can take in a gameplaying setting. We provide a summary of motivations and some indicative examples for each of the four AI uses for gameplaying.

game (RPG) such *The Elder Scrolls V: Skyrim* (Bethesda Softworks, 2011) has only one player role, the rest are non-player characters. In general, non-player roles have more limited possibilities than player roles.

In summary, AI could be playing a game to **win** or for the **experience** of play either by taking the role of the **player** or the role of a **non-player** character. This yields four core uses of AI for playing games as illustrated in Fig. 3.1. With these distinctions in mind, we will now look at these four key motivations for building game-playing AI in further detail.

### 3.1.1 Playing to Win in the Player Role

Perhaps the most common use of AI together with games in academic settings is to play to win, while taking the role of a human player. This is especially common when using games as an AI testbed. Games have been used to test the capabilities and performance of AI algorithms for a very long time, as we discussed in Section 1.2. Many of the milestones in AI and games research have taken the form of some sort of AI program beating the best human player in the world at some games. See, for example, IBM's Deep Blue winning over Garry Kasparov in Chess, Google DeepMind's AlphaGo winning over Lee Sedol [629] and Ke Jie in Go, and IBM's

Watson winning Jeopardy! [201]. All of these were highly publicized events widely seen as confirmations of the increasing capabilities of AI methods. As discussed in Chapter 1, AI researchers are now increasingly turning to video games to find appropriate challenges for their algorithms. The number of active competitions associated with the IEEE CIG and AIIDE conferences is testament to this, as is DeepMind’s and Facebook AI Research’s choice of *StarCraft II* (Blizzard Entertainment, 2015) as a testbed for their research.

Games are excellent testbeds for artificial intelligence for a number of reasons, as elaborated on in Section 1.3. An important reason is that games are made to test human intelligence. Well-designed games exercise many of our cognitive abilities. Much of the fun we have in playing games comes from learning the games through playing them [351], meaning that well-designed games are also great teachers. This, in turn, means that they offer the kind of gradual skill progression that allows for testing of AI at different capability levels.

There are some reasons besides AI benchmarking for why you might want to use an AI in the place of a human to play games to win. For example, there are some games where you need strong AI to provide a challenge to players. This includes many strategic games of perfect information, such as classic board games, including Chess, Checkers and Go. However, for games with hidden information, it is often easier to provide challenge by simply “cheating”, for example, by giving the AI player access to the hidden state of the game or even by modifying the hidden state so as to make it harder to play for the human. For example, in the epic strategy game *Civilization* (MicroProse, 1991), all civilizations can be played by human players. However, playing any Civilization game well under the same conditions as a human is very challenging, and there is, to our knowledge, no AI capable of playing these games as well as a good human player. Therefore, when playing against several computer-controlled civilizations, the game typically cheats by providing these with preferential conditions in various ways.

Another use case for AI that plays to win in a player role is to test games. When designing a new game, or a new game level, you can use a game-playing agent to test whether the game or level is playable, so called **simulation-based testing**. However, in many cases you want the agent to also play the game in a human-like manner to make the testing more relevant; see below on playing for experience.

Historically, the use of AI to play to win in a player role has been so dominant in academic work that some researchers have not even considered other roles for AI in playing games. In game development, on the other hand, this particular motivation for game-playing AI is much more rare; most game-playing AI in existing games is focused on non-player roles and/or playing for experience. This mismatch has historically contributed to the lack of understanding between academia and industry on game AI. In recent years however, there has been a growing understanding of the multitude of motivations for game-playing AI.

### 3.1.2 Playing to Win in a Non-player Role

Non-player characters are very often designed to *not* offer maximum challenge or otherwise be as effective as possible, but instead to be entertaining or human-like; see below for non-player characters playing for experience. However, there are instances when you want a non-player character to play as well as possible. As mentioned above, strategy games such as *Civilization* (MicroProse, 1991) have an (unanswered) need for high-performing non-cheating opponents, though here we are talking about playing roles that other human players could in principle have taken. Other strategy games, such as *XCOM: Enemy Unknown* (2K Games, 2012), have playing roles that humans would not play, creating a need for NPC AI playing to win.

Other times, creating an NPC playing to win is a necessary precursor to creating an NPC playing for experience. For example, in a racing game, you might want to implement “rubber band AI” where the NPC cars adapt their speed to the human player, so that they are never too far behind or ahead. Doing this is easy, but only if you already have an AI controller that can play the game well, either through actually playing the game well or through cheating in a way that cannot easily be detected. The performance of the controller can then be reduced when necessary so as to match the player’s performance.

### 3.1.3 Playing for Experience in the Player Role

Why would you want an agent that takes the role of a human player, but that does not focus on winning? For example, when you want a **human-like agent**. Perhaps the most important reason for such agents is alluded to above: simulation-based testing. This is important both when designing games and game content manually, and when generating content procedurally; in the latter case, the quality of the game content is often evaluated automatically with the help of an agent playing the game, as discussed in Chapter 4. When trying to see how the game would be played by a human it is therefore important that the agent plays in a human-like manner, meaning that it has performance comparable to a human, has similar reaction speed, makes the same sort of mistakes that a human would do, is curious about and explores the same areas as a human would, etc. If the AI agent plays significantly differently from how a human would play, it might give the wrong information about e.g., whether a game is winnable (it might be winnable but only if you have superhuman reflexes) or whether a game mechanic is used (maybe a human would use it, but not an AI that tries to play optimally).

Another situation where human-like play is necessary is when you want to demonstrate how to play a level to a human player. A common feature of games is some kind of **demo mode**, which shows the game in action. Some games even have a demonstration feature built into the core gameplay mode. For example, *New Super Mario Bros* (Nintendo, 2006) for the Nintendo Wii will show you how to play

a particular part of a level if you fail it repeatedly. The game simply takes over the controls and plays for you for about 10 to 20 seconds, and lets you continue afterwards. If all the level content is known beforehand, and there are no other players, such demonstrations can be hardcoded. If some parts of the game are user-designed, or procedurally generated, the game needs to generate these demonstrations itself.

Playing in a “human-like” fashion may seem a rather fuzzy and subjective aim, and it is. There are many ways in which a typical AI agent plays differently from a typical human player. How humans and AIs differ depends on the algorithm used to play the game, the nature of the game itself, and a multitude of other factors. To investigate these differences further, and spur the development of agents that can play in a human-like manner, two different Turing test-like competitions have been held. The 2K BotPrize was held from 2008 to 2013, and challenged competitors to develop agents that could play the FPS *Unreal Tournament 2004* (Epic Games, 2004) in such a way that human participants thought that the bots were human [263, 262, 647]. Similarly, the Turing test track of the Mario AI Competition let people submit playing agents of *Super Mario Bros* (Nintendo, 1985), who were judged by human onlookers as to whether they were human or not [619, 717]. While it takes us too far to go through all of the results of these competitions here, there are some very obvious signs of non-humanness that recur across games for many types of AI agents. These include having extremely fast reactions, switching between actions faster than a human could, not attempting actions which fail (because of having a too good model of the outcome of actions), not doing unnecessary actions (such as jumping when one could just be running) and not hesitating or stopping to think.

Of course, not all players of a game play in the same way. In fact, as discussed further in Chapter 5, if one analyzes a set of play traces of any game one can often find a number of player “archetypes” or “personas”, clusters of players who play the game in markedly different ways in terms of e.g., aggression, speed, curiosity and skill. Within work on AI that plays games in human-like styles, there has been work both on learning and mirroring the playstyle of individual players [422, 423, 511, 328, 603] and on learning to play games in the style of one of several personas [267, 269].

### 3.1.4 Playing for Experience in a Non-player Role

Almost certainly the most common goal for game-playing AI in the game industry is to make non-player characters act, almost always in ways which are not primarily meant to beat the player or otherwise “win” the game (for many NPCs it may not even be defined what winning the game means). NPCs may exist in games for many, sometimes overlapping, purposes: to act as adversaries, to provide assistance and guidance, to form part of a puzzle, to tell a story, to provide a backdrop to the action of the game, to be emotively expressive and so on [724]. The sophistication and behavioral complexity of NPCs likewise vary widely, from the regular left-right movements of the aliens in *Space Invaders* (Midway, 1978) and Koopas in the *Super*

*Mario Bros* (Nintendo, 1985–2016) series to the nuanced and varied behavior of non-player characters in *Bioshock Infinite* (2K Games, 2013) and the alien in *Alien: Isolation* (Sega, 2014).

Depending on the role of the NPC, very different tasks can be asked of the AI algorithms that control it. (It can certainly be argued that many of the scripts that control NPCs cannot truthfully be described as *artificial intelligence* in any conventional way, but we will stick with the acronym here as it is commonly used for all code that controls non-player characters in the game industry.) In many cases what the game designers look for is **the illusion of intelligence**: for the player to believe that the NPC in some sense is intelligent even though the code controlling it is very simple. Human-likeness in the sense discussed in the previous section might or might not be the objective here, depending on what kind of NPC it is (a robot or a dragon should perhaps not behave in a too human-like manner).

In other cases, the most important feature of an NPC is its **predictability**. In a typical stealth game, a large part of the challenge is for the player to memorize and predict the regularities of guards and other characters that should be avoided. In such cases, it makes sense that the patrols are entirely regular, so that their schedule can be gleaned by the player. Similarly, the *boss monsters* in many games are designed to repeat certain movements in a sequence, and are only vulnerable to the player’s attack when in certain phases of the animation cycle. In such cases, too “intelligent” and adaptive behavior would be incompatible with the game design.

It should be noted that even in cases where you would expect to need supple, complex behavior from NPCs, an agent that plays to win might be very problematic. Many high-performing strategies are seen as very **boring** by the player, and prime examples of “unsportsmanlike” behavior. For example, in an experiment with building high-performing AI for a turn-based strategy game, it was found that one of the solutions (based on neuroevolution) was extremely boring to play against, as it simply took a defensive position and attacked any incoming units with long-distance attacks [490]. Similarly, *camping* (staying stationary in a protected position and waiting for enemies to expose themselves to fire) is a behavior that is generally frowned on and often banned in FPS games, but it is often highly effective and easy to learn for an AI (incidentally, real-life military training often emphasizes camping-like tactics—what is effective is often not fun). Another interesting example is the work by Denzinger et al. [165] in which an evolutionary algorithm found that the best way to score a goal in *FIFA 99* (Electronic Arts, 1999) was by forcing a penalty kick. The evolutionary process found a local optimum in the fitness landscape corresponding to a *sweet spot* or *exploit* of the game’s mechanics which yielded highly efficient, yet predictable and boring gameplay. Exploiting the game’s bugs for winning is a *creative* strategy that is not only followed by AIs but also by human players [381].

### 3.1.5 Summary of AI Game-Playing Goals and Roles

We argued above that playing to win in the player role has been overemphasized, to the point of neglecting other perspectives, in much of academic research. In the same way, work on AI in the game industry has generally overemphasized playing for experience in a non-player role, to the point of neglecting other perspectives. This has led to an emphasis in the industry on behavior authoring methods such as finite state machines and behavior trees, as AI methods based on search, optimization and learning have been seen as not conducive to playing for experience; a common gripe has been a perceived lack of predictability and authorial control with such methods. However, given a better understanding of what roles AI can be used to play in games, this neglect of methods and perspectives is hopefully coming to an end in both academia and industry.

## 3.2 Game Design and AI Design Considerations

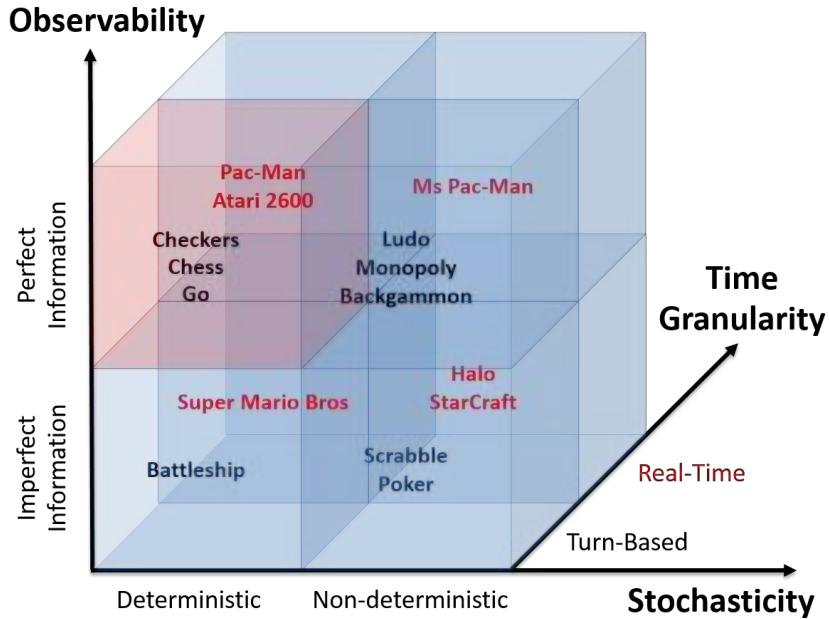
When choosing an AI method for playing a particular game (in any of the roles discussed in Section 3.1) it is crucial to know the characteristics of the game you are playing and the characteristics of the algorithms you are about to design. These collectively determine what type of algorithms can be effective. In this section we first discuss the challenges we face due to the characteristics of the game per se (Section 3.2.1) and then we discuss aspects of AI algorithmic design (Section 3.2.2) that need to be considered independently of the game we examine.

### 3.2.1 Characteristics of Games

In this section we discuss a number of characteristics of games and the impact they have on the potential use of AI methods. All characteristics covered are tied to the design of the game but a few (e.g., input representation and forward model) are also dependent on the technical implementation of the game and possibly amenable to change. Much of our discussion is inspired by the book *Characteristics of Games* by Elias et al. [192], which discusses many of these factors from a game design perspective. For illustrative purposes, Fig. 3.2 places a number of core game examples onto the three-dimensional space of **observability**, **stochasticity** and **time granularity**.

#### 3.2.1.1 Number of Players

A good place to start is the number of players a game has. Elias et al. [192] distinguish between:



**Fig. 3.2** Characteristics of games: game examples across the dimensions of stochasticity, observability and time granularity. Note that the game examples presented are sorted by complexity (action space and branching factor) within each cube. Minimax can *theoretically* solve merely any deterministic, turn-based game of perfect information (red cube in the figure)—in practice, it is still impossible to solve games with substantially large branching factors and action spaces such as Go via Minimax. Any AI method that eventually approximates the Minimax tree (e.g., MCTS) can be used to tackle imperfect information, non-determinism and real-time decision making (see blue cubes in figure). Strictly speaking, *Super Mario Bros* (Nintendo, 1985) involves a small degree of non-determinism only when a player helps creating a particular scene; we can, thus, safely classify the game as deterministic [163].

- **single-player** games, such as puzzles and time-trial racing;
- **one-and-a-half-player** games, such as the campaign mode of an FPS with non-trivial NPCs;
- **two-player** games, such as Chess, Checkers and *Spacewar!* (Russell, 1962); and
- **multi-player** games, such as *League of Legends* (Riot Games, 2009), the *Mario Kart* (Nintendo, 1992–2014) series and the online modes of most FPS games.

The distinction between single-player and one-and-a-half-player games is not a sharp one—there is no clear boundary for how advanced NPCs should be to count as a “half player”. In the case of multi-player games, many can be played with only two players, at which point they are effectively two-player games. It is not always the case that other players (or NPCs) are adversarial and try to stop the player—there are many collaborative games, or games where relations between players are complex and have elements of both competition and cooperation. Still, keeping the

number of players in mind is very useful when thinking about algorithms for playing games.

When using tree search algorithms to play games, some algorithms fit particularly well with some numbers of players. Standard single-agent tree search algorithms such as breadth-first, depth-first and A\* fit the single-player case particularly well (including games which have NPCs, but where those NPCs are so simple and predictable as to be treated as part of the environment). In such games, what happens in the game is determined entirely by the actions the player takes and any potential random effects; there are no other “intentional” players. This fits very well with single-agent tree search algorithms, which are based on the **Markov property**, that the next state is entirely determined by the previous state and the action taken at that point.

A particular case is **two-player zero-sum adversarial games**, i.e., there are exactly two players; one player will win, the other will lose (or perhaps there will be a draw). We do not know what the other player will do, but we can safely assume that she will do everything she can to win, and thereby deny you the victory. The Minimax algorithm (with or without  $\alpha$ - $\beta$  pruning) is perfectly suited to this case, and will lead to optimal play given sufficient computation time.

But how do we cope with the challenge when we have **many players**, or perhaps a single player surrounded by very complicated non-player agents? While it is theoretically possible to expand Minimax to multiple players, this only works if there can be no collusion (or alliances of any kind) between players and the zero-sum nature of the game still remains (which it usually does not). Further, the computational complexity of Minimax quickly gets unmanageable with more than two players, as for every move you take you do not just need to consider the countermove of one player, but of all players. So this approach is rarely workable.

It is more common in the multi-player case to treat the game as a single-player game, but use some kind of model of what the other players do. This could be an assumption that the other players will oppose the player, a learned model based on observed behavior or even a random model. With an appropriate model of what the other players will do, many standard single-player game-playing methods can be used in multi-player settings.

### 3.2.1.2 Stochasticity

A common way in which many games violate the Markov property is by being **stochastic** (or non-deterministic). In many games, some of what happens is random. As standard digital computer architectures do not allow for “true” randomness, effective randomness is provided by pseudo-random number generators. The word “stochastic” is used to denote processes which cannot be practically predicted, whether they result from true randomness or complex calculations. Games can have varying amounts of stochasticity, from completely deterministic games like Chess to games dominated by stochastic outcomes like Roulette, Ludo, Yahtzee or even Monopoly. It is common for games to have mostly or fully deterministic game

mechanics combined with some stochastic element through card-drawing, dice-throwing or some similar mechanism to reduce the possibility of planning. However, stochasticity can occur in essentially any part of a game.

In a game with stochasticity, the outcome of the game is not entirely determined by the actions the players take. In other words, if you play several playthroughs of the same game, taking the same actions at the same points in time, you are not guaranteed the same outcome. This has consequences for AI algorithms. For **tree search** algorithms, it means that we cannot be sure about the state which a sequence of actions will lead to, and therefore about the results of the algorithm. This leads to problems with using many tree search algorithms in their canonical forms, and requires that we add some modifications to address the non-deterministic uncertainty in the forward model. For example, in Monte Carlo tree search modifications such as **determinization** are used, where the different possible outcomes of each action are explored separately [77]. While these algorithm variations can be effective, they generally increase the computational complexity of the base algorithm.

For **reinforcement learning** approaches, including evolutionary reinforcement learning, it means that we have reduced certainty in exactly how good a given strategy/policy is—a good policy may achieve bad outcomes, or a bad policy good outcomes, because of random events in the game. Such outcome uncertainty can be mitigated by evaluating every policy multiple times, though this has significant computational cost. On the other hand, stochasticity can sometimes actually be an advantage when learning policies: a policy which is learned for a stochastic game may be more robust than one learned for a deterministic game, as in the latter case it is possible to learn a very brittle policy that only works for a specific configuration of the game. For example, learning a policy which attacks enemies in specific places at specific times, rather than being able to handle enemies that might arrive from any direction at any time.

While it is very common for digital games to include some form of stochasticity, an interesting case is very early games hardware such as the 1977-vintage Atari 2600, which does not have the facilities for implementing pseudo-random number generators (mainly because it lacks a system clock). If a player takes exactly the same actions at exactly the same times (including the key press that starts the game), exactly the same outcome will be achieved. The Arcade Learning Environment is a widely used game-based AI-benchmark built around an emulator of the Atari 2600 [40]. When training AI agents to play games with no stochasticity, it is entirely possible to learn brittle policies that effectively bypass the complexities of the full game (whether this actually happens, or whether most agents learn more general strategies, is an open question). As we already saw across the various examples of Chapter 2 *Ms Pac-Man* (Namco, 1982) is arguably the most popular non-deterministic arcade game of that era.

### 3.2.1.3 Observability

**Observability** is a characteristic that is strongly related to stochasticity. It refers to how much information about the game state is available to the player(s). At one extreme we have classic board games such as Chess, Go and Checkers, where the full board state is always available to the players, and puzzles such as Sudoku and Spelltower. These games have **perfect information**. At the other extreme we can think of classic text adventures such as *Zork* (Personal Software, 1980) or *Colossal Cave Adventure*, where initially very little of the world and its state is revealed to the player and much of the game is about exploring what the world is like. Those games have **hidden information** and therefore only **partial observability**. Many, if not most, computer games have significant hidden information: think of a typical platform game such as *Super Mario Bros* (Nintendo, 1985), or FPS such as the *Halo* series (Microsoft Studios, 2001–2015), where at any point you can only perceive a small part of the game world. Within computer strategy games such as *StarCraft II* (Blizzard Entertainment, 2015) or *Civilization* (MicroProse, 1991) the common term for hidden information is *fog of war*. Even many classic non-digital games have hidden information, including most card games where players keep their hands of cards private (such as Poker) and board games such as *Battleship*.

When developing an AI agent for a game with hidden information, the simplest approach you can follow is to merely ignore the hidden information. At each point in time, just feed the available information to the agent and use that to decide the next action. Doing so actually works quite well in some games, notably action-focused games with linear levels; for example, simple *Super Mario Bros* (Nintendo, 1985) levels can be played well based on only instantaneously available information [706]. However, if you play a strategy game such as *StarCraft* (Blizzard Entertainment, 1998) based on only the available information, you are not even going to see the enemy until it is too late—good play involves active information gathering. Even in *Super Mario Bros* (Nintendo, 1985), complicated levels that feature backtracking require remembering off-screen parts of the level [322]. In a trick-taking card game such as Poker the available information (your own hand) is actually of comparatively less relevance; the core of the game is modeling the hidden information (your adversaries' hands and minds).

Therefore, effective AI for games with partial observability often requires some kind of **modeling of the hidden information**. For some games, notably variants of Poker such as Heads-up limit hold'em, considerable research has been done on game-specific methods for modeling hidden information that includes opponents' play [63]. There are also more generic methods of adding some form of hidden state modeling to existing algorithms, such as Information Set Monte Carlo tree search [146]. Just like when it comes to methods for dealing with stochasticity, these methods typically add considerable computational complexity compared to the base algorithm.

### 3.2.1.4 Action Space and Branching Factor

When you play the minimalist-masochist mobile game *Flappy Bird* (dotGEARS, 2013), you have a single choice at any point in time: to flap or not to flap. (Flapping is accomplished by touching the screen, and makes the protagonist bird rise in the air.) *Flappy Bird* and similar one-button games, such as *Canabalt* (Beatshapers, 2009), probably have the lowest possible **branching factor**. The branching factor is the number of different actions you can take at any decision point. The branching factor of Flappy Bird is 2: flap or no flap.

For comparison, *Pac-Man* (Namco, 1980) has a branching factor of 4: up, down, left and right. *Super Mario Bros* (Nintendo, 1985) has a branching factor of around 32: eight D-pad directions times two buttons (though you may argue that some of these combinations are nonsensical and should not actually be considered). Chess has an average branching factor of 35, whereas Checkers has a somewhat lower branching factor. Go has a whooping 400 for the very first move; as the board is populated, the branching factor decreases, but there are typically a few hundred potential positions to put every stone.

While 400 is a very high branching factor compared to 35 or 2, it dwarfs in comparison to many computer strategy games where multiple units can be moved every turn. Considering each combination of movements of individual units as an action, this means that the branching factor of the game is the product of the branching factor of the individual units. If you have 6 different units that can each take 10 different actions at a given time—a rather conservative estimate compared to typical games of, say, *StarCraft* (Blizzard Entertainment, 1998) or *Civilization* (Micro-Prose, 1991)—then your branching factor is a million!

But wait, it gets worse. For many games, it is not even possible to enumerate all the actions, as the input space is **continuous**. Think of any modern first-person game played on a computer or console. While it is true that computers do not really capture infinities well, and that “continuous” inputs such as computer mice, touchscreens and thumbsticks (on e.g., XBox and Playstation controllers) actually return a digital number, that number has such fine resolution that it is for all practical purposes continuous. The only way to create a practically enumerable set of actions is to discretize the continuous input space somehow, and strike a compromise between overwhelming branching factors and reducing the input space so much as to not be able to play the game effectively.

The branching factor is a key determinant of the effectiveness of **tree search** algorithms. The complexity of the breadth-first algorithm (for single-player games) and the Minimax algorithm (for adversarial two-player games) for searching to depth  $d$  is  $b^d$ , where  $b$  is the branching factor. In other words, a high branching factor makes it almost impossible to search more than a few steps ahead. This fact has very tangible consequences for which games can be played with tree search methods; for example, Go has an order of magnitude higher branching factor than Chess, and this was arguably the main reason for the very poor performance of all kinds of AI methods on Go for decades (during which the same methods performed well on Chess). Monte Carlo tree search handles high branching factors better be-

cause it builds imbalanced trees, but it is by no means immune to the problem. Once the branching factor gets high enough (say, a million, or maybe a billion, depending on the speed of the simulator), it becomes impractical to enumerate even the actions at depth 1 and therefore to use tree search at all.

High branching factors are an issue for **reinforcement learning** algorithms as well, including evolutionary reinforcement learning. Mostly this has to do with the controller/policy representation. If you are using a neural network (or some other function approximator) for representing your policy, you may need to have outputs for each action; alternatively, if you are using the network to assign values to all actions, you need to iterate over them. In both cases, a large number of possible actions carries a cost. Another problem is the exploration-exploitation dilemma: the higher the number of possible actions, the longer it will take to explore them all while learning.

A final comment on branching factors is that for many games, they are not constant. In Chess, you have fewer moves available at the beginning of the game when most of your pieces are blocked, more towards the midgame, and fewer again in the endgame when most pieces may be blocked. In a typical RPG such as those in the *Final Fantasy* series (Square Enix, 1987–2016) the number of available actions increases as the player character accrues items, spells and other possibilities. As mentioned above, the number of available actions in Go decreases as you play.

### 3.2.1.5 Time Granularity

When discussing branching factors above, we talked about the number of possible actions to take at any “point in time”. But how often is that? How often can the player take an action? A fundamental distinction is that between **turn-based** and **real-time** games. Most classic board games are turn-based games. In such games, players take turns, and at each turn a player can take an action, or a specified number of actions. The amount of real time that passes between turns is generally not of any importance inside the game (though tournaments and professional play often incorporate some form of time limit). Real-time games include many popular genres of computer games, such as FPS, racing games and platformers. Even within real-time games, there is considerable variation in how often an in-game action can in practice be taken. At the extreme there is the screen update frequency; the current generation of video games typically strives to have an update frequency of 60 frames per second to ensure a perceived smooth movement, but many games update the screen half as often or even less because of the complexity of rendering complicated scenes. In practice, the number of actions a player character (or any other in-game character) could take per second is usually more limited than that.

To take two examples far apart on the time granularity scale, let us consider two adversarial games: Chess and *StarCraft* (Blizzard Entertainment, 1998). A game of Chess between skilled players on average lasts about 40 turns.<sup>1</sup> In *StarCraft* (Bliz-

---

<sup>1</sup> <http://chess.stackexchange.com/questions/2506/>

zard Entertainment, 1998), a highly competitive real-time strategy (RTS) game, professional players often take three to five actions per second (each action is typically executed with a mouse click or a shortcut key). With a typical game lasting 10 to 20 minutes, this means that thousands of actions are taken in a game. But there are not that many more significant events in a game of *StarCraft* (Blizzard Entertainment, 1998) than in a game of Chess—the lead does not change much more often, and grand strategic decisions are not made that much more often. This means that the number of actions between significant game events is much higher in *StarCraft* (Blizzard Entertainment, 1998) than in Chess.

Time granularity affects AI game-playing methods through limiting how far ahead you can look. A given **depth** of search means very different things depending on the time granularity of the game. Ten turns in Chess is enough to execute a whole strategy; ten actions ahead in *StarCraft* (Blizzard Entertainment, 1998) might just be a few seconds, during which the game might not have changed in any significant way. To play *StarCraft* (Blizzard Entertainment, 1998) well using tree search, one would need an exceptional search depth, in the hundreds or thousands of actions, which would clearly be computationally infeasible. One way to address this challenge is to consider **macro-actions** (e.g., as in [525, 524]), which are sets or sequences of smaller, fine-grained, actions.

### 3.2.2 Characteristics of AI Algorithm Design

In the following, we discuss some important issues in applying AI algorithms to games. These are design choices relating not so much to game design (covered in the previous section), as to AI algorithm design and the constraints under which the algorithm is used. This section expands the discussion about representation and utility covered in Chapter 2 with a focus on game-playing AI.

#### 3.2.2.1 How Is the Game State Represented?

Games differ in what information they present to the player, and how. Text adventures output text, the state of a classic board game can be described by the positions of all board pieces, and graphical video games serve moving graphics together with sound and occasionally outputs such as controller rumble. For digital games, the technical limitations of the hardware on which the game is implemented influences how it is presented; as processor speed and memory capacity increases, the pixel resolution and scene complexity of video games has increased commensurably.

Importantly, the same game can be represented in different ways, and which way it is represented matters greatly to an algorithm playing the game. To take a racing game as an example, the algorithm could receive a first-person view out of the windscreen of the car rendered in 3D, or an overhead view of the track rendering the track and various cars in 2D. It could also simply receive a list of positions and

velocities of all cars on the track in the frame of reference of the track (along with a model of the track), or a set of angles and distances to other cars (and track edges) in the frame of reference of the track.

The choices regarding input representation matter a lot when designing a game. If you want to learn a policy for driving a car around a track, and the inputs to the policy are the three continuous variables associated with speed and distance to the left and right edge of the track, learning a decent driving policy is comparatively simple. If your input is instead an unprocessed visual feed—i.e., tens of thousands of pixel values—finding a decent policy is likely to be much harder. Not only is the policy search space in the latter case vastly greater, the proportion of the search space that corresponds to decently-performing policies is likely to be much smaller, as many more nonsensical policies are possible (e.g., turn left if even-numbered pixels are lighter than odd-numbered pixels; this policy does not map to the game state in any sensible way, and if it works it is a fluke). In order to learn to drive well based on visual input—at least in cases where illumination, roadside scenery, etc. vary significantly—you likely need to learn a visual system of some kind. In light of this, most naive policies applied to full visual input would likely not have fared very well. Continuing the car racing example, even in cases where you have very few inputs, how these are represented matters; for example, it is much easier to learn a good driving policy if the inputs are represented in the frame of reference of the car rather than that of the track [707, 714]. A somewhat more comprehensive discussion on ways of representing low-dimensional inputs to neural networks can be found in [567].

In recent years, several groups of researchers have focused on learning policies that use full visual feeds as inputs. For example, Koutnik et al. evolved neural networks to play *The Open Racing Car Simulator* (TORCS) from high-resolution video [353], Kempka et al. used the pixels of the screen as input to a deep Q network that was trained to play a version of *DOOM* (GT Interactive, 1993) [333], and Mnih et al. trained deep networks to play Atari 2600 games using Q-learning [464]. Using the raw pixel inputs is often motivated by giving the AI the same conditions as a human would have, and thus achieving a level playing field between human and AI. Another motivation is that if you want to use your algorithm to play a game “out of the box”, without any API or additional engineering to expose the internal state of the game, you will likely have to resort to using the raw visual feed. However, in cases where you have access to the source code of the game or a useful API—as you would almost always have when developing AI for a new game—there is no reason to not utilize the “digested” game state in whatever form makes the task of the AI algorithm easiest. Whether or not to present information that the human player does not have access to, i.e., “cheating”, is a separate question.

### 3.2.2.2 Is There a Forward Model?

A very important factor when designing an AI to play a game is whether there is a simulator of the game, a so-called **forward model**, available. A forward model is a

model which, given a state  $s$  and an action  $a$ , reaches the same state  $s'$  as the real game would reach if it was given  $a$  at  $s$ . In other words, it is a way of playing the game in simulation, so that consequences of multiple actions can be explored before actually taking some of those actions in the real game. Having a forward model of the game is necessary in order to be able to use any tree search-based approaches to playing a game, as those approaches depend on simulating the outcome of multiple actions.

A very desirable property of a forward model, in addition to that it exists, is that it is **fast**. In order to be able to use a tree search algorithm effectively for control in a real-time game, one would generally need to be able to simulate gameplay at least a thousand times faster than real-time, preferably tens or hundreds of thousands of times faster.

It is very easy to construct a forward model for classic board games such as Chess and Go, as the game state is simply the board state and the rules are very easy to encode. For many video games, constructing a forward model can be done by simply copying (or otherwise reusing) the same code as is used for controlling the game itself, but without waiting for user input or displaying graphics, and without performing all the calculations involved with graphics rendering. For some video games—notably games that were originally implemented for much older hardware, such as classic arcade games that were implemented on 8-bit or 16-bit processors—forward models can be made much faster than real-time, as the core game loop is not that computationally complex. (It might also be possible to do this with some modern games, by running them inside emulators that can replace the graphics routines with dummy code.)

For many games, however, it is impossible or at least very hard to obtain a fast forward model. For most commercial games, the source code is not available, unless you are working at the company that develops the game. Even if the source code is available, current software engineering practices in the game industry make it very hard to extract forward models from game code, as the core control loops are often closely tied up with user interface management, rendering, animation and sometimes network code. A change in software engineering practices to separate the core game loop more cleanly from various input/output functions so that forward models could more easily be built would be one of the most important enablers of advanced AI methods in video games. However, in some cases the computational complexity of the core game loop might still be so high that any forward models built on the core game code would be too slow to be usable. In some of such cases, it might be practical to build and/or learn a simplified or **approximate forward model**, where the state resulting from a series of actions taken in the forward model is not guaranteed to be identical to the state resulting from the same series of actions in the actual game. Whether an approximate forward model is acceptable or not depends on the particular use case and motivation for the AI implementation. Note that a somewhat less than accurate forward model might still be desirable. For example, when there is significant hidden information or stochasticity the AI designer might not want to provide the AI agent with an *oracle* that makes the hidden information observable

and tells the agent which random actions will happen. Taking such a design decision might lead to unreasonably good performance and the appearance of cheating.

When a forward model cannot be produced, **tree search** algorithms cannot be applied. It is still possible to manually construct agents, and also to learn agents through **supervised learning** or some form of **reinforcement learning**, such as temporal difference learning or evolutionary reinforcement learning. However, note that while the reinforcement learning approaches in general do not need a complete forward model in the sense that the results of taking any action in any state can be predicted, they still need a way to run the game faster than real-time. If the game cannot be sped up significantly beyond the pace at which it is naturally played, it is going to take the algorithm a very long time to learn to play.

### 3.2.2.3 Do You Have Time to Train?

A crude but useful distinction in artificial intelligence is between algorithms that try to decide **what to do in a given situation** by examining possible actions and future states—roughly, tree search algorithms of various kinds—and algorithms that **learn a model** (such as a policy) over time—i.e., machine learning. The same distinction exists within AI for playing games. There are algorithms developed that do not need to learn anything about the game, but do need a forward model (tree search); there are algorithms that do not need a forward model, but instead learn a policy as a mapping from state(s) to action (model-free reinforcement learning); and there are algorithms that require both a forward model and training time (model-based reinforcement learning and tree search with adaptive hyperparameters).

What type of algorithm you will want to use depends largely on your motivation for using AI to play games. If you are using the game as a testbed for your AI algorithm, your choice will be dictated by the type of algorithm you are testing. If you are using the AI to enable player experience in a game that you develop—for example, in a non-player role—then you will probably not want the AI to perform any learning while the game is being played, as this risks interfering with the gameplay as designed by the designer. In other cases you are looking for an algorithm that can play some range of games well, and do not have time to retrain the agent for each game.

### 3.2.2.4 How Many Games Are You Playing?

An aim the AI designer might wish to achieve is that of **general game playing**. Here, we are not looking for a policy for a single game, we are looking for a more generic agent that can play any game that it is presented with—or at least any game from within a particular distribution or genre, and which adheres to a given interface. General game playing is typically motivated by a desire to use games to progress towards artificial *general* intelligence, i.e., developing AI that is not only good at one thing but at many different things [598, 679, 744]. The idea is to avoid overfitting

(manually or automatically) a given game and come up with agents that generalize well to many different games; it is a common phenomenon that when developing agents for a particular game, for example, for a game-based AI competition, many special-purpose solutions are devised that do not transfer well to other games [701].

For this reason, it is common to evaluate general game playing agents on unseen games, i.e., games on which they have not been trained and which the designers of the agent were not aware of when developing the agent. There are several frameworks for general game playing, including the General Game Playing Competition [223], the General Video Game AI Competition [528, 527] and the Arcade Learning Environment [40]. These will be discussed later in the chapter.

General video game playing, where AI is developed to play for performance in the player role across many games (ideally all games), can be seen as diametrically opposed to the typical use of AI for playing games in commercial game development, where the AI is playing for experience in a non-player role, and is carefully tuned to a particular game. However, the development of AI methods for general game playing certainly benefits commercial game AI in the end. And even when developing game AI as part of game development, it is good engineering practice to develop methods that are reusable to some extent.

## 3.3 How Can AI Play Games?

In Chapter 2, we reviewed a number of important AI methods. Most of these methods can be used to play games in one way or another. This section will focus on the core AI methods, and for each family of algorithms it will go through on how they can be used to play games.

### 3.3.1 Planning-Based Approaches

Algorithms that select actions through planning a set of future actions in a state space are generally applicable to games, and do not in general require any training time. They do require a fast forward model if searching in the game's state space, but not if simply using them for searching in the physical space (path-planning). Tree search algorithms are widely used to play games, either on their own or in supporting roles in game-playing agent architectures.

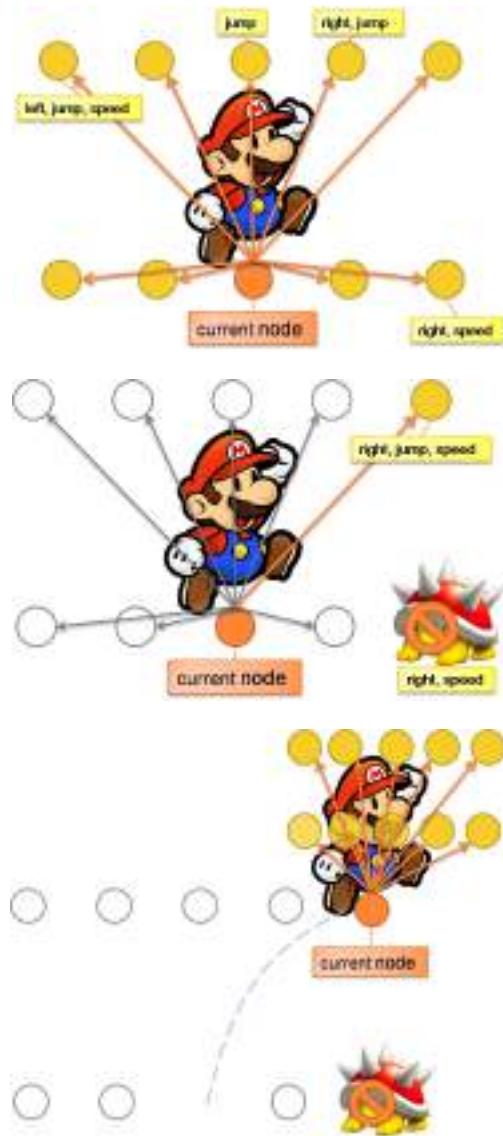
#### 3.3.1.1 Classic Tree Search

Classic tree search methods, which feature little or no randomness, have been used in game-playing roles since the very beginning of research on AI and games. As mentioned in the introduction of this book the Minimax algorithm and  $\alpha$ - $\beta$  pruning

were originally invented in order to play classic board games such as Chess and Checkers [725]. While the basic concepts of adversarial tree search have not really changed since then, there have been numerous tweaks to existing algorithms and some new algorithms. In general, classic tree search methods can easily be applied in games that feature full observability, a low branching factor and a fast forward model. Theoretically they can solve any deterministic game that features full observability for the player (see the red cube in Fig. 3.2); in practice, they still fail in games containing large state spaces.

Best-first search, in particular a myriad variations of the A\* algorithm, is very commonly used for **path-planning** in modern video games. When an NPC in a modern 3D FPS or RPG decides how to get from point A to point B, this is typically done using some version of A\*. In such instances, search is usually done in (in-game) physical space rather than state space, so no forward model is necessary. As the space is pseudo-continuous, search is usually done on the nodes of a mesh or lattice overlaid on the area to be traversed. Note that best-first search is only used for navigation and not for the full decision-making of the agent; methods such as behavior trees or finite-state machines (which are usually hand-authored) are used to determine *where* to go, whereas A\* (or some variation of it) is used to determine *how* to get there. Indeed, in games where player input is by pointing to and clicking at positions to go—think of an overhead brawler like *Diablo* (Blizzard Entertainment, 1996) or an RTS like *StarCraft* (Blizzard Entertainment, 1998)—the execution of the player’s order usually involves a path-planning algorithm as well. Recent additions to the family of best-first algorithms include jump point search (JPS), which can improve performance by orders of magnitude compared to standard A\* under the right circumstances [662]. Hierarchical pathfinding is its own little research area based on the idea of dividing up an area into subareas and using separate algorithms for deciding how to go between and within the areas. Choosing a path-planning algorithm for a modern video game is usually a matter of choosing the algorithm that works best given the shape of the environments the NPCs (or PCs) are traversing, the particular way a grid or movement graph is overlaid on top of this space, and the demands of the animation algorithm. Generally, one size does not fit all; some textbooks devoted to industry-oriented game AI discuss this in more depth [461].

Beyond path-planning, best-first algorithms such as A\* can be used for **controlling** all aspects of NPC behavior. The key to doing this is to search in the state space of the game, not just the physical space. (Obviously, this requires a fast forward model.) To take an example, the winner of the 2009 Mario AI Competition was entirely based on A\* search in state space [705]. This competition tasked competitors with developing agents that could play a Java-based clone of the classic platform game *Super Mario Bros* (Nintendo, 1985)—it later evolved into a multi-track competition [322]. While a forward model was not supplied with the original competition software, the winner of the competition, Robin Baumgarten, created one by adapting parts of the core game code. He then built an A\* agent which at any point simply tried to get to the right edge of the screen. (An illustration of the agent can be seen in Figs. 3.3 and 2.5.) This worked extremely well: the resulting agent played seemingly optimally, and managed to get to the end of all levels in-



**Fig. 3.3** An illustration of the key steps of A\* search for playing *Super Mario Bros* (Nintendo, 1985). The agent considers a maximum of nine possible actions at each frame of the game, as a result of combining the jump and speed buttons with moving right or left (top figure). Then the agent picks the action with the highest heuristic value (middle figure). Finally, the Mario agent takes the action (i.e., right, jump, speed in this example), moves to a new state and evaluates the new action space in this new state (bottom figure). More details about the A\* agent that won the Mario AI competition in 2009 can be found in [705].

cluded in the competition software. A video showing the agent navigating one of the levels gathered more than a million views on YouTube;<sup>2</sup> the appeal of seeing the agent playing the game is partly the extreme skill of the agent in navigating among multiple enemies.

It is important to note that the success of this agent is due to several factors. One is that the levels are fairly linear; in a later edition of the competition, levels with dead ends which required back-tracking were introduced, which defeated the pure A\* agent [322]. Two other factors are that *Super Mario Bros* (Nintendo, 1985) is deterministic and has locally perfect information (at any instant the information in the current screen is completely known) and of course that a good forward model is available: if the A\* would not have used a complete model of the game including the movement of enemies, it would have been impossible to plan paths around these enemies.

### 3.3.1.2 Stochastic Tree Search

The MCTS algorithm burst onto the scene of Go research in 2006 [141, 77], and heralded a quantum leap in performance of Go-playing AI. Classic adversarial search had performed poorly on Go, partly because the branching factor is too high (about an order of magnitude higher than Chess) and partly because the nature of Go makes it very hard to algorithmically judge the value of a board state. MCTS partly overcomes these challenges by building imbalanced trees where not all moves need to be explored to the same depth (reduces effective branching factor) and by doing random rollouts until the end of the game (reduces the need for a state evaluation function). The AlphaGo [629] software which beat two of the best human Go players in the world in 2016 and 2017, is built around the MCTS algorithm.

The success of MCTS on Go has led researchers and practitioners to explore its use for playing a wide variety of other games, including trading card games [746], platform games [294], real-time strategy games [311, 645], racing games [203] and so on. Of course, these games differ in many ways from Go. While Go is a deterministic perfect information game, a real-time strategy game such as *StarCraft* (Blizzard Entertainment, 1998), a trading card game such as *Magic: The Gathering*, or any Poker variant feature both hidden information and stochasticity. Methods such as Information Set Monte Carlo tree search are one way of dealing with these issues, but impose computational costs of their own [146].

Another problem is that in games with fine time granularity, it might take a prohibitively long time for a rollout to reach a terminal state (a loss or a win); in many video games it is possible to take an arbitrary number of actions without winning or losing the game, or even doing something that materially affects the outcome of the game. For example, in *Super Mario Bros* (Nintendo, 1985), most randomly generated action sequences would not see Mario escaping the original screen, but basically pacing back and forth until time runs out, thousands of time steps later.

---

<sup>2</sup> <https://www.youtube.com/watch?v=DlkMs4ZHHr8>

One response to this problem is to only roll out a certain number of actions, and if a terminal state is not encountered use a state evaluation function [77]. Other ideas include pruning the action selection so as to make the algorithm search deeper [294]. Given the large number of modifications to all components of the MCTS algorithm, it makes more sense to think of MCTS as a general algorithmic framework rather than as a single algorithm.

Many games could be played either through MCTS, uninformed search (such as breadth-first search) or informed search (such as A\*). Deciding which method to use is not always straightforward, but luckily these methods are relatively simple to implement and test. Generally speaking, Minimax can only be used for (two-player) adversarial games whereas other forms of uninformed search are best used for single-player games. Best-first search requires some kind of estimate of a distance to a goal state, but this does not need to be a physical position or the end goal of the game. Varieties of MCTS can be used for both single-player and two-player games, and often outperform uninformed search when branching factors are high.

### 3.3.1.3 Evolutionary Planning

Interestingly, decision making through planning does not need to be built on tree search. Alternatively, one can use optimization algorithms for planning. The basic idea is that instead of searching for a sequence of actions starting from an initial point, you can optimize the whole action sequence. In other words, you are searching the space of complete action sequences for those that have maximum utility. Evaluating the utility of a given action sequence is done by simply taking all the actions in the sequence in simulation, and observing the value of the state reached after taking all those actions.

The appeal of this idea is that an optimization algorithm might search the plan space in a very different manner compared to a tree search algorithm: all tree search algorithms start from the root of the tree (the origin state) and build a tree from that point. Evolutionary algorithms instead regard the plan as simply a sequence, and can perform mutations or crossover at any point in the string. This could help in guiding the search at different areas of the plan space that a tree search algorithm would explore for the same problem.

While many different optimization algorithms could be used, the few studies on optimization-based planning in games that can be found in the literature use evolutionary algorithms. Perez et al. proposed using evolutionary planning for single-player action games, calling this approach “rolling horizon evolution” [526]. In the particular implementation for the Physical Traveling Salesman Problem (a hybrid between the classic TSP problem and a racing game), an evolutionary algorithm was used to generate a plan every time step. The plan was represented as a sequence of 10-20 actions, and a standard evolutionary algorithm was used to search for plans. After a plan was found, the first step of the plan was executed, just as would be the case with a tree search algorithm. Agents based on evolutionary planning generally perform competitively in the General Video Game AI Competition [528].

Evolutionary planning is particularly promising as a technique for handling very large branching factors, as we have seen that games with multiple independent units (such as strategy games) can have. Justesen et al. [309] applied evolutionary computation to select actions in the turn-based strategy game *Hero Academy* (Robot Entertainment, 2012), calling this approach “online evolution”. Given the number of units the player controls and the number of actions available per unit, the branching factor is about one million; therefore, only a single turn ahead was planned. Evolutionary planning was shown to outperform Monte Carlo tree search by a wide margin in that game. Wang et al. [745], and Justesen and Risi [310] later applied variants of this technique to *StarCraft* (Blizzard Entertainment, 1998) tactics. Given the continuous-space nature of the game, the branching factor would be extreme if every possible movement direction for every unit was considered as a separate action. What was evolved was therefore not a sequence of actions, but rather which of several simple scripts (tactics) each unit would use in a given time step (this idea was borrowed from Churchill and Buro, who combined a “portfolio” of scripts with simple tree search [123]). Wang et al. [745] showed that evolutionary planning performed better than several varieties of tree search algorithms in this simple *StarCraft* (Blizzard Entertainment, 1998) scenario.

Evolutionary planning in games is a recent invention, and there are only a limited number of studies on this technique so far. It is not well understood under what conditions this technique performs well, or even really why it performs so well when it does. A major unsolved problem is how to perform evolutionary adversarial planning [586]; whereas planning based on tree search works in the presence of an adversary (for example, see the minimax algorithm), it is not clear how to integrate this into a genotype. Perhaps through competitive coevolution of actions taken by different players? There is, in other words, plenty of scope for further research in this area.

### 3.3.1.4 Planning with Symbolic Representations

While planning on the level of in-game actions requires a fast forward model, there are other ways of using planning in games. In particular, one can plan in an abstract representation of the game’s state space. The field of automated planning has studied planning on the level of symbolic representations for decades [228]. Typically, a language based on first-order logic is used to represent events, states and actions, and tree search methods are applied to find paths from the current state to an end state. This style of planning originated with the STRIPS representation used in Shakey, the world’s first digital mobile robot [494]; symbolic planning has since been used extensively in numerous domains.

The horror-themed first-person shooter *F.E.A.R.* (Sierra Entertainment, 2005) became famous within the AI community for its use of planning to coordinate NPC behavior. The game’s AI also received nice reviews in the gaming press, partly because the player is able to hear the NPCs communicate with each other about their plan of attack, heightening immersion. In *F.E.A.R.* (Sierra Entertainment, 2005),

a STRIPS-like representation is used to plan which NPCs perform which actions (flank, take cover, suppress, fire, etc.) in order to defeat the player character. The representation is on the level of individual rooms, where movement between one room and the next is usually a single action [507]. Using this high-level representation, it is possible to plan much further ahead than would be possible when planning on the scale of individual game actions. Such a representation, however, requires manually defining states and actions.

### 3.3.2 Reinforcement Learning

As discussed in Chapter 2, a reinforcement learning algorithm is any algorithm that solves a reinforcement learning problem. This includes algorithms from the temporal difference or approximate dynamic programming family (for simplicity, we will refer to such algorithms as *classic* reinforcement learning methods), applications of evolutionary algorithms to reinforcement learning such as neuroevolution and genetic programming, and other methods. In this section, we will discuss both classic methods (including those that involve deep neural networks) and evolutionary methods as they are applied for playing games. Another way of describing the difference between these methods is the difference between *ontogenetic* (which learns during “lifetimes”) and *phylogenetic* (which learns between “lifetimes”) methods [715].

Reinforcement learning algorithms are applicable to games when there is learning time available. Usually this means plenty of training time: most reinforcement learning methods will need to play a game thousands, or perhaps even millions, of times in order to play it well. Therefore, it is very useful to have a way of playing the game much faster than real-time (or a very large server farm). Some reinforcement learning algorithms, but not all, also require a forward model. Once it has been trained, a reinforcement-learned policy can usually be executed very fast.

It is important to note that the planning-based methods (described in the previous section) for playing games cannot be directly compared with the reinforcement learning methods described in this section. They solve different problems: planning requires a forward model and significant time at each time step; reinforcement learning instead needs learning time and may or may not need a forward model.

#### 3.3.2.1 Classic and Deep Reinforcement Learning

As already mentioned in the introduction of this book, classic reinforcement learning methods were used with games early on, in some cases with considerable success. Arthur Samuel devised an algorithm—which can be said to be the first classic reinforcement learning algorithm—in 1959 to create a self-learning Checkers player. Despite the very limited computational resources of the day, the algorithm learned to play well enough to beat its creator [591]. Another success for classic reinforcement learning in game-playing came a few decades later, when Gerald

Tesauro used the modern formulation of temporal difference learning to teach a simple neural network to play Backgammon, named TD-gammon; it learned to play surprisingly well, after starting with no information and simply playing against itself [689] (TD-gammon is covered in more detail in Chapter 2). This success motivated much interest in reinforcement learning during the 1990s and early 2000s.

However, progress was limited by the lack of good function approximators for the value function (e.g., the Q function). While algorithms such as Q-learning will provably converge to the optimal policy under the right conditions, the right conditions are in fact very restrictive. In particular, they include all state values or {state, action} values that are stored separately, for example, in a table. However, for most interesting games there are far too many possible states for this to be feasible—almost any video game has at least billions of states. This means that the table would be too big to fit in memory, and that most states would never be visited during learning. It is clearly necessary to use a compressed representation of the value function that occupies less memory and also does not require every state to be visited in order to calculate its value. It can, instead, calculate it based on neighboring states that have been visited. In other words, what is needed is a **function approximator**, such as a neural network.

However, using neural networks together with temporal difference learning turns out to be non-trivial. It is very easy to encounter “catastrophic forgetting”, where sophisticated strategies are unlearned in favor of degenerate strategies (such as always taking the same action). The reasons for this are complex and go beyond the discussion in this chapter. However, to intuitively understand one of the mechanisms involved, consider what would usually happen for a reinforcement learning agent playing a game. Rewards are very sparse, and the agent will typically see long stretches of no reward, or negative reward. When the same reward is encountered for a long time, the backpropagation algorithm will be trained only with the target value of that reward. In terms of supervised learning, this is akin to training for a long term on a single training example. The likely outcome is that the network learns to only output that target value, regardless of the input. More details on the method of approximating a value function using an ANN can be found in Section 2.8.2.3.

A major success in the use of reinforcement learning of the temporal difference variety together with function approximators came in 2015, when Google Deep-Mind published a paper where they managed to train deep neural networks to play a number of different games from the classic Atari 2600 games console [464]. Each network was trained to play a single game, with the inputs being the raw pixels of the game’s visuals, together with the score, and the output being the controller’s directions and fire button. The method used to train the deep networks is deep Q networks, which is essentially standard Q-learning applied to neural networks with many layers (some of the layers used in the architecture were convolutional). Crucially, they managed to overcome the problems associated with using temporal difference techniques together with neural networks by a method called *experience replay*. Here, short sequences of gameplay are stored, and replayed to the network in varying order, in order to break up the long chains of similar states and reward.

This can be seen as akin to batch-based training in supervised learning, using small batches.

### 3.3.2.2 Evolutionary Reinforcement Learning

The other main family of reinforcement learning methods is evolutionary methods. In particular, using evolutionary algorithms to evolve the weights and/or topology of neural networks (**neuroevolution**) or programs, typically structured as expression trees (**genetic programming**). The fitness evaluation consists in using the neural network or program to play the game, and using the result (e.g., score) as a fitness function.

This basic idea has been around for a long time, but was surprisingly under-explored for a long time. John Koza, a prominent researcher within genetic programming, used an example of evolving programs for playing Pac-Man in his 1992 book [356]. A couple of years later, Pollack and Blair showed that evolutionary computation can be used to train backgammon players using the same setup as Tesauro used in his experiments with TD learning, and with similar results [537]. Outside of games, a community of researchers was forming in the 1990s exploring the idea of using evolutionary computation to learn control strategies for small robots; this field came to be called **evolutionary robotics** [496]. Training robots to solve simple tasks of e.g., navigation, obstacle-avoidance and situational learning has very much in common with training NPCs to play games, in particular two-dimensional arcade-like games [567, 767, 766].

Starting around 2005, a number of advances were made in applying neuroevolution to playing different types of video games. This includes applications to car racing [707, 709, 392, 353], first-person shooters [518], strategy games [79], real-time strategy games [654] and classic arcade games such as Pac-Man [766, 403]. Perhaps the main takeaway from this work is that neuroevolution is extremely versatile, and can be applied to a wide range of games, usually in several different ways for each game. For example, for a simple car racing game it was shown that evolving neural networks that acted as state evaluators, even in combination with a simple one-step lookahead search, substantially outperformed evolving neural networks working as action evaluators (Q functions) [408]. Input representation matters too; as discussed in Section 3.2.2.1, egocentric inputs are generally strongly preferred, and there are additional considerations for individual game types, such as how to represent multiple adversaries [654].

Neuroevolution has seen great success in learning policies in cases where the state can be represented using relatively few dimensions (say, fewer than 50 units in the neural network’s input layer), and is often easier to tune and get working than classic reinforcement learning algorithms of the temporal difference variety. However, neuroevolution seems to have problems scaling up to problems with very large input spaces that require large and deep neural networks, such as those using high-dimensional pixel inputs. The likely reason for this is that stochastic search in weight space suffers from the *curse of dimensionality* in a way that gradient descent

search (such as backpropagation) does not. Currently, almost all successful examples of learning directly from high-dimensional pixel inputs use deep Q-learning or similar methods, though there are approaches that combine neuroevolution with unsupervised learning, so that controllers are learned that use a compressed representation of the visual feed as input [353].

For more details on the general method of neuroevolution and pointers to the literature the reader is referred to Section 2.8.1, and to the recent survey paper on neuroevolution in games [567].

### **3.3.3 Supervised Learning**

Games can also be played using **supervised learning**. Or rather, policies or controllers for playing games can be learned through supervised learning. The basic idea here is to record traces of human players playing a game and train some function approximator to behave like the human player. The traces are stored as lists of tuples  $\langle \text{features}, \text{target} \rangle$  where the features represent the game state (or an observation of it that would be available to the agent) and the target is the action the human took in that state. Once the function approximator is adequately trained, the game can be played—in the style of the human(s) it was trained on—by simply taking whatever action the trained function approximator returns when presented with the current game state. Alternatively, instead of learning to predict what action to take, one can also learn to predict the value of states, and use the trained function approximator in conjunction with a search algorithm to play the game. Further details about the potential supervised algorithms that can be used in games are described in Chapter 2.

### **3.3.4 Chimeric Game Players**

While planning, reinforcement learning and supervised learning are fundamentally different approaches to playing games, solving the game-playing problem under different constraints, which does not mean that they cannot be combined. In fact, there are many examples of successful **hybrids** or **chimeras** of approaches from these three broad classes. One example is **dynamic scripting** [650] which can be viewed as a form of a **learning classifier system** [363] in that it involves a rule-based (here called script-based) representation coupled with reinforcement learning. Dynamic scripting adjusts the importance of scripts via reinforcement learning at runtime and is based on the current game state and immediate rewards obtained. Dynamic scripting has seen several applications in games including fighting games [417] and real-time strategy games [409, 154]. The approach has been used mainly for AI that adapts to the skills of the player, thereby aiming at the experience of the player and not necessarily at winning the game.

Another good example is AlphaGo. This extremely high-performing Go-playing agent actually combines planning through search, reinforcement learning and supervised learning [629]. At the core of the agent is a Monte Carlo tree search algorithm which searches in the state space (planning). Rollouts, however, are combined with evaluations from a neural network which estimates the value of states, and node selection is informed by a position estimation network. Both the state network and position network are initially trained on databases of games between grandmasters (supervised learning), and later on further trained by self-play (reinforcement learning).

## 3.4 Which Games Can AI Play?

Different games pose different challenges for AI playing, in the same way they pose different challenges for human playing. Not only are there differences in what kind of access the AI player has to the games, but also between different game types: a policy for playing Chess is unlikely to be proficient at playing the games in the *Grand Theft Auto* (Rockstar Games, 1997–2013) series. This section is organized according to game genres, and for each game genre it discusses what the particular cognitive, perceptual, behavioral and kinesthetic challenges games of that genre generally pose, and then gives an overview of how AI methods have been used to play that particular game genre. It also includes several extended examples, giving some detail about particular implementations. Once again it is important to note that the list is not inclusive of all possible game genres AI can play as a player or non-player character; the selection is made on the basis of popularity of game genres and the available published work on AI for playing games in each genre.

### 3.4.1 Board Games

As discussed in the introduction of this book, the earliest work on AI for playing games was done in classic board games, and for a long time that was the only way in which AI was applied to playing games. In particular, Chess was so commonly used for AI research that it was called the “*drosophila of artificial intelligence*” [194], alluding to the use of the common fruit fly as a model organism in genetics research. The reasons for this seem to have been that board games were simple to implement, indeed possible at all to implement on the limited computer hardware available in the early days of AI research, and that these games were seen to require something akin to “pure thought”. What a game such as Chess or Go does require is **adversarial planning**. Classic board games typically place no demand at all on perception, reactions, motor skills or estimation of continuous movements, meaning that their skill demands are particularly narrow, especially compared to most video games.

Most board games have very simple discrete state representations and deterministic forward models—the full state of the game can often be represented in less than 100 bytes, and calculating the next state of the game is as simple as applying a small set of rules—and reasonably small branching factors. This makes it very easy to apply **tree search**, and almost all successful board game-playing agents use some kind of tree search algorithm. As discussed in the sections on tree search in Chapter 2, the Minimax algorithm was originally invented in the context of playing Chess. Decades of research concentrated on playing Chess (with a lesser amount of research on Checkers and Go), with specific conferences dedicated to this kind of research, led to a number of algorithmic advances that improved the performance of the Minimax algorithm on some particular board game. Many of these have limited applicability outside of the particular game they were developed on, and it would take us too far to go into these algorithmic variations here. For an overview of advances in Chess playing, the reader is referred to [98].

In Checkers the reigning human champion was beaten by the Chinook software in 1994 [594] and the game was *solved* in 2007, meaning that the optimal set of moves for both players was found (it is a draw if you play optimally) [593]; in Chess, Garry Kasparov was famously beaten by Deep Blue in 1997 [98]. It took until 2016 for Google DeepMind to beat a human Go champion with their AlphaGo software [629], mainly because of the algorithmic advances necessary. Whereas Chess and Checkers can be played effectively with some variation of the Minimax algorithm combined with relatively shallow state evaluations, the larger branching factor of Go necessitated and spurred the development of MCTS [77].

While MCTS can be utilized to play board games without a state evaluation function, supplementing that algorithm with state and action evaluation functions can massively enhance the performance, as seen in the case of AlphaGo, which uses deep neural networks for state and action evaluation. On the other hand, when using some version of Minimax it is necessary to use state evaluation functions as all interesting board games (more complex than Tic-Tac-Toe) have too large state spaces to be searched until the end of the game in acceptable time. These evaluation functions can be manually constructed, but in general it is a very good idea to use some form of learning algorithm to learn their parameters (even though the structure of the function is specified by the algorithm designer). As discussed above, Samuel was the first to use a form of reinforcement learning to learn a state evaluation function in a board game (or any kind of game) [591], and Tesauro later used TD learning to very good effect in Backgammon [689]. Evolutionary computation can also be used to learn evaluation functions, for example, Pollack showed that co-evolution could perform well on Backgammon using a very similar setup to Tesauro [537]. Noteworthy examples of strong board game players based on evolved evaluation functions are *Blondie24* [207] and *Blondie25* [208], a Checkers- and a Chess-playing program respectively. The evaluation functions were based on five-layered deep convolutional networks, and *Blondie25* in particular performed well against very strong Chess players.

While classic board games such as Go and Chess have existed for hundreds or even thousands of years, the past few decades have seen a rejuvenation of board

game design. Many of the more recently designed board games mix up the formula of classic board games with design thinking from other game genres. A good example is *Ticket to Ride* (Days of Wonder, 2004), which is a board game that includes elements of card games, such as variable numbers of players, hidden information and stochasticity (in the draw of the cards). For these reasons, it is hard to construct well-performing AI players based on standard tree-search methods; the best known agents include substantial domain knowledge yet perform poorly compared to human players [160]. Creating generic well-performing agents for this type of game is an interesting research challenge.

Given the simplicity of using tree search for board game playing, it is not surprising that every approach we have discussed so far builds on one tree search algorithm or another. However, it is possible to play board games without forward models—usually with results that are “interesting” rather than good. For example, Stanley and Miikkulainen developed a “roving eye” approach to playing Go, where an evolved neural network self-directedly scans the Go board and decides where to place the next piece [656]. Relatedly, it is reportedly possible for the position evaluation network of AlphaGo to play a high-quality game of Go on its own, though it naturally plays stronger if combined with search.

### 3.4.2 Card Games

**Card games** are games centered on one or several decks of cards; these might or might not be the standard 52-card French deck which is commonly used in classic card games. Most card games involve players possessing different cards that change ownership between players, or between players and the deck, or other positions on the table. Another important element of most card games is that some cards are visible to the player who possesses them but not to other players. Therefore, almost all card games feature a large degree of **hidden information**. In fact, card games are perhaps the type of games where hidden information most dominates gameplay.

For example, take the classic card game *Poker*, which is currently very popular in its *Texas hold 'em* variety. The rules are relatively simple: the player which at the end of a few rounds holds the best cards (the “best hand”) wins. Between the rounds, the player can exchange a number of cards for fresh cards drawn from the deck. If there were perfect information, i.e., all players could see each others’ hands, this would be an uninteresting game that could be played according to a lookup table. What makes Texas hold ‘em—and similar Poker variants—challenging and interesting is that each player does not know what cards the other players have. The cognitive challenges of playing these games involve acting in the absence of information, which implies inferring the true game state from incomplete evidence, and potentially affecting other players’ perception of the true game state. In other words, a game of Poker is largely about guessing and bluffing.

A key advance in playing Poker and similar games is the Counterfactual Regret Minimization (CFR) algorithm [797]. In CFR, algorithms learn by self-play in a

similar fashion to how temporal difference learning and other reinforcement learning algorithms have been used in perfect information games like Backgammon and Checkers. The basic principle is that after every action, when some hidden state has been revealed, it computes the alternative reward of all other actions that could have been taken, given the newly revealed information. The difference between the reward attained from the action that was actually taken and the best action that could have been taken is called the *regret*. The policy is then adjusted so as to minimize the regret. This is done iteratively, slowly converging on a policy that is optimal in the sense that it loses as little as possible over a large number of games. However, for games as complex as Texas hold 'em, simplifications have to be done in order to use the CFR algorithm in practice.

DeepStack is a recent agent and algorithm that has reached world-class performances in Texas hold 'em [467]. Like CFR, DeepStack uses self-play and recursive reasoning to learn a policy. However, it does not compute an explicit strategy before play. Instead, it uses tree search in combination with a state value approximation to select actions at each turn. In this sense, it is more like the heuristic search of AlphaGo (but in a setting with plenty of imperfect information) than like the reinforcement-learned policy of TD-gammon.

Another, much more recent, card game which is drawing increasing interest from the research community is *Hearthstone* (Blizzard Entertainment, 2014); see Fig. 3.4. This is a collectible card game in the tradition of *Magic: The Gathering*, but with somewhat simpler rules and only played on computers. A game of Hearthstone takes place between two players, with each player having a deck of 30 cards. Each card represents either a creature or a spell. Each player has a handful of cards ( $< 7$ ) in hand (invisible for the other player), and at each turn draws a new card and has the option of playing one or more cards. Creature cards convert to creatures that are placed on the player's side of the table (visible for both players), and creatures can be used to attack the opponent's creatures or player character. Spells have a multiplicity of different effects. The hundreds of different cards in the game, the possibility of choosing to take multiple actions each turn, the long time taken to play a game (20 to 30 turns is common), the presence of stochasticity and of course the hidden information (mainly what cards are in the opponent's hand) conspire to make *Hearthstone* (Blizzard Entertainment, 2014) a hard game to play for both humans and machines.

Perhaps the simplest approach to playing *Hearthstone* (Blizzard Entertainment, 2014) is to simply ignore the hidden information and play each turn in a greedy fashion, i.e., search the space of possible actions within a single turn and choose the one that optimizes some criterion such as health point advantage at the end of that turn, given the available information only. Agents that implement such greedy policies are included with some open source Hearthstone simulators, such as *Metastone*.<sup>3</sup> Standard tree search algorithms such as Minimax or MCTS are generally ineffective here (as in Poker) because of the very high degree of hidden information. One approach to constructing high-performing agents is instead to hand-code

---

<sup>3</sup> <http://www.demilich.net/>



**Fig. 3.4** A screenshot from *Hearthstone* (Blizzard Entertainment, 2014) displaying a number of different creature or spell cards available in the game. Image obtained from Wikipedia (fair use).

domain knowledge, for example, by building an ontology of cards and searching in an abstract symbolic space [659].

Unlike in Poker, where the player has no control over what cards it is dealt, in *Hearthstone* (Blizzard Entertainment, 2014) the player can also construct a deck with which to play the game. This adds another level of challenge to playing the game: in addition to choosing what action(s) to take at each turn, the successful player must also construct what allows her to implement her strategy. The composition of the deck effectively constrains what strategy can be chosen, and then implemented tactically through action selection. While these two levels interplay—a strong player takes the composition of the deck and the strategy it affords into account when choosing a move, and vice versa—it is also true that the problems of deck building and action selection can to some extent be treated separately, and implemented in different agents. One approach to deck building is to use evolutionary computation. The deck is seen as the genome, and the fitness function involves simple heuristic agents using the deck for playing [218]. A similar approach has also been used in the multi-player card game *Dominion* [416].

### 3.4.3 Classic Arcade Games

Classic arcade games, of the type found in late 1970s and early 1980s arcade cabinets, home video game consoles and home computers, have been commonly used as AI benchmarks within the last decade. Representative platforms for this game type



(a) *Track & Field* (Konami, 1983) is a game about athletics. The game screenshot depicts the start of the 100 m dash. Image obtained from Wikipedia (fair use).

(b) In *Tapper* (Bally Midway, 1983) the player controls a bartender who serves drinks to customers. Image obtained from Wikipedia (fair use).

**Fig. 3.5** *Track & Field* (Konami, 1983), *Tapper* (Bally Midway, 1983), and most classic arcade games require rapid reactions and precision.

are the Atari 2600, Nintendo NES, Commodore 64 and ZX Spectrum. Most classic arcade games are characterized by movement in a two-dimensional space (sometimes represented isometrically to provide the illusion of three-dimensional movement), heavy use of graphical logics (where game rules are triggered by intersection of sprites or images), continuous-time progression, and either continuous-space or discrete-space movement.

The cognitive challenges of playing such games vary by game. Most games require fast reactions and precise timing, and a few games, in particular early sports games such as *Track & Field* (Konami, 1983) and *Decathlon* (Activision, 1983), rely almost exclusively on speed and reactions (Fig. 3.5(a)). Very many games require prioritization of several co-occurring events, which requires some ability to predict the behavior or trajectory of other entities in the game. This challenge is explicit in e.g., *Tapper* (Bally Midway, 1983)—see Fig. 3.5(b)—but also in different ways part of platform games such as *Super Mario Bros* (Nintendo, 1985), shooting galleries such as *Duck Hunt* (Nintendo, 1984) or *Missile Command* (Atari Inc., 1980) and scrolling shooters such as *Defender* (Williams Electronics–Taito, 1981) or *R-type* (Irem, 1987). Another common requirement is navigating mazes or other complex environments, as exemplified most clearly by games such as *Pac-Man* (Namco, 1980), *Ms Pac-Man* (Namco, 1982), *Frogger* (Sega, 1981) and *Boulder Dash* (First Star Software, 1984), but also common in many platform games. Some games, such as *Montezuma's Revenge* (Parker Brothers, 1984), require long-term planning involving the memorization of temporarily unobservable game states. Some games feature incomplete information and stochasticity, others are completely deterministic and fully observable.

### 3.4.3.1 Pac-Man and Ms Pac-Man

Various versions and clones of the classic *Pac-Man* (Namco, 1981) game have been frequently used in both research and teaching of artificial intelligence, due to the depth of challenge coupled with conceptual simplicity and ease of implementation. In all versions of the game, the player character moves through a maze while avoiding pursuing ghosts. A level is won when all pills distributed throughout the level are collected. Special power pills temporarily give the player character the power to consume ghosts rather than being consumed by them. As seen in Chapter 2, the differences between the original *Pac-Man* (Namco, 1981) and its successor *Ms Pac-Man* (Namco, 1982) may seem minor but are actually fundamental; the most important is that one of the ghosts in *Ms Pac-Man* (Namco, 1982) has non-deterministic behavior, making it impossible to learn a fixed sequence of actions as a solution to the game. The appeal of this game to the research community is evidenced by a recent survey covering over 20 years of active AI research using these two games as testbeds [573].

Several frameworks exist for Pac-Man-based experimentation, some tied to competitions. The Pac-Man screen capture competition is based around the Microsoft Revenge of Arcade version of the original game, and does not provide a forward model nor facilities for speeding up the game [404]. The Ms Pac-Man vs Ghost Team competition framework is written in Java and includes both a forward model and ability to speed up the game significantly; it also includes an interface for controlling the ghost team rather than Ms Pac-Man, the player character [574]. The Atari 2600 version of *Ms Pac-Man* (Namco, 1982) is available as part of the ALE framework.<sup>4</sup> There is also a Python-based Pac-Man framework used for teaching AI at UC Berkeley.<sup>5</sup>

As expected, the performance of AI players varies depending on the availability of a forward model, which allows the simulation of ghost behavior. The screen capture-based competition, which does not offer a forward model, is dominated by heuristic approaches (some of them involve pathfinding in the maze without taking ghost movement into account), which perform at the level of beginner human players [404]. It has been observed that even searching one step ahead, and using a state evaluator based on an evolved neural network, can be an effective method for playing the game [403]. Of course, searching deeper than a single ply yields additional benefits; however, the stochasticity introduced in *Ms Pac-Man* (Namco, 1982) poses challenges even in the presence of a forward model. MCTS has been shown to work well in this case [590, 524]. Model-free approaches to reinforcement learning have also been used for playing the game with some success [57]. In general, the best competitors in the Ms Pac-Man vs Ghost Team Competition play at the level of intermediate-skill human players [574]. At the moment of writing this book *Ms Pac-Man* (Namco, 1982) is reported to be practically solved (reaching the maximum possible score of 999,990 points) by the Microsoft Maluuba team. The

<sup>4</sup> [www.arcadelearningenvironment.org](http://www.arcadelearningenvironment.org)

<sup>5</sup> [http://ai.berkeley.edu/project\\_overview.html](http://ai.berkeley.edu/project_overview.html)

team used an RL technique called hybrid reward architecture [738] which decomposes the reward function of the environment into different RL problems (a set of reward functions) that a corresponding number of agents need to solve. Each agent selects its actions by considering the aggregated Q values for each action across all agents.

*Pac-Man* (Namco, 1980) can also be played for experience rather than performance. In a series of experiments, neural networks that controlled the ghosts in a clone of the game were evolved to make the game more entertaining for human players [766]. The experiment was conceptually based on Malone’s definition of fun in games as challenge, curiosity and fantasy dimensions [419] and sought to find ghost behavior that maximized these traits. In particular, the fitness was composed of three factors: 1) the appropriate level of challenge (i.e., when the game is neither too hard nor too easy), 2) the diversity of ghost behavior, and 3) the ghosts’ spatial diversity (i.e., when ghosts behavior is explorative rather than static). The fitness function used to evolve interesting ghost behaviors was cross-validated via user studies [770].

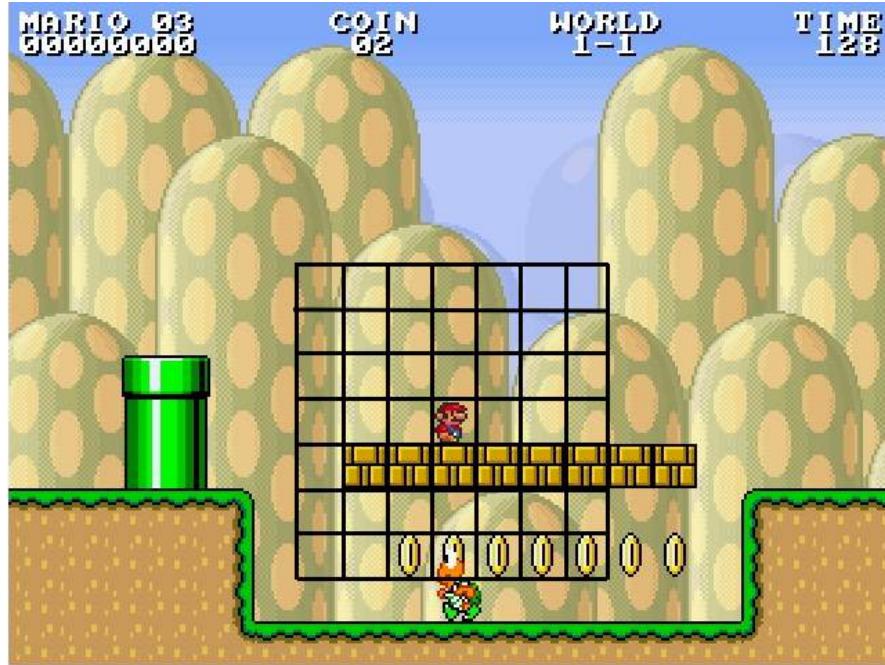
### 3.4.3.2 Super Mario Bros

Versions and clones of Nintendo’s landmark platformer *Super Mario Bros* (Nintendo, 1985) have been extremely popular for AI research, including research on game playing, content generation and player modeling (research using this game is described in several other parts of this book). A large reason for this is the *Mario AI Competition*, which was started in 2009 and included several different tracks focused on playing for performance, playing in a human-like manner and generating levels [322, 717]. The software framework for that competition<sup>6</sup> was based on *Infinite Mario Bros* (Notch, 2008), a Java-based clone of *Super Mario Bros* (Nintendo, 1985) featuring simple level generation [706, 705]. Different versions of the competition software, generally referred to as the Mario AI Framework or Mario AI Benchmark, have since been used in many dozens of research projects. In the following, we will for simplicity refer to methods for playing various versions of *Super Mario Bros* (Nintendo, 1985), *Infinite Mario Bros* or the Mario AI Framework/Benchmark simply as playing “Mario”.

The first version of the Mario AI could be simulated thousands of times faster than real-time, but did not include a forward model. Therefore the first attempts to learn a Mario-playing agent was through learning a function from a state observation directly to Mario actions [706]. In that project, neural networks were evolved to guide Mario through simple procedurally generated levels. The inputs were the presence or absence of environment features or enemies in a coarse grid centered on Mario, and the outputs were interpreted as the button presses on the Nintendo controller (up, down, left, right). See Fig. 3.6 for an illustration of the state representation. A standard feedforward MLP architecture was used for the neural network,

---

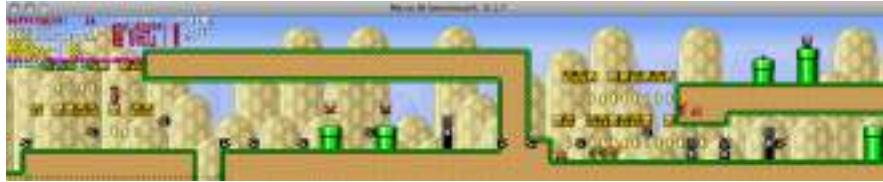
<sup>6</sup> <http://julian.togelius.com/mariocompetition2009/>



**Fig. 3.6** The inputs to the Mario-playing neural network are structured as a Moore neighborhood centered on Mario. Each input is 1 if the corresponding tile is occupied by a tile (such as ground) that Mario cannot pass through, and 0 otherwise. In another version of the experiment, a second set of inputs was added where an input was 1 if there was an enemy at the corresponding tile. Image adapted from [706].

and the fitness function was simply how far the controller was able to progress on each level. Using this setup and a standard evolution strategy, neural networks were evolved that could win some levels but not all, and generally played at the strength of a human beginner.

However, as is so often the case, having a forward model makes a big difference. The first Mario AI Competition, in 2009, was won by Robin Baumgarten, who constructed a forward model for the game by reusing some of the open-source game engine code [705]. Using this model, he constructed an agent based on A\* search in state space. At each time frame, the agent searches for the shortest path towards the right edge of the screen, and executes the first action in the resulting plan. As the search utilizes the forward model and therefore takes place in state space rather than just physical space, it can incorporate the predicted movements of the (deterministic) enemies in its planning. This agent was able to finish all the levels used in the 2009 Mario AI Competition, and produces behavior that appears optimal in terms of time to complete levels (it does not focus on collecting coins or killing enemies). See Section 2.3.2 for an explanation of the algorithm and a figure illustrating its use in Mario.



**Fig. 3.7** An example level generated for the 2010 Mario AI Competition. Note the overhanging structure in the middle of the screenshot, creating a dead end for Mario; if he chooses to go beneath the overhanging platform, he will need to backtrack to the start of the platform and take the upper route instead after discovering the wall at the end of the structure. Agents based on simple A\* search are unable to do this.

Given the success of the A\*-based agent in the 2009 competition, the next year's edition of the competition updated the level generator so that it generated more challenging levels. Importantly, the new level generator created levels that included "dead ends", structures where Mario can take the wrong path and if so must backtrack to take the other path [322]. See Fig. 3.7 for an example of such a dead end. These structures effectively "trap" agents that rely on simple best-first search, as they end up searching a very large number of paths close to the current position, and time out before finding a path that backtracks all the way to beginning of the structure. The winner of the 2010 Mario AI Competition was instead the REALM agent [56]. This agent uses an evolved rule-based system to decide sub-goals within the current segment of the level, and then navigates to these sub-goals using A\*. REALM successfully handles the dead ends that were part of the levels in the 2010 competition, and is as far as we know the highest-performing Mario-playing agent there is.

Other search algorithms beyond A\* have been tried for playing Mario, including Monte Carlo tree search [294]. It was found that the standard formulation of MCTS did not perform very well, because the algorithm did not search deep enough and because the way the average reward of a branch is calculated results in risk-averse behavior. However, with certain modifications to remedy these problems, an MCTS variant was found that could play Mario as well as a pure A\* algorithm. In a follow-up experiment, noise was added to the Mario AI Benchmark and it was found that MCTS handled this added noise much better than A\*, probably because MCTS relies on statistical averaging of the reward whereas A\* assumes a deterministic world.

All of the above work has been focused on playing for performance. Work on playing Mario for experience has mostly focused on imitating human playing styles, or otherwise creating agents that play Mario similarly to a human. To further this research, a Turing test track of the Mario AI Competition was created [619]. In this track, competitors submitted agents, and their performance on various levels was recorded. Videos of the agents playing different levels were shown to human spectators along with videos of other humans playing the same levels, and the spectators were asked to indicate which of the videos were of a human player. Agents were scored based on how often they managed to fool humans, similarly to the setup of

the original Turing test. The results indicated simple heuristic solutions that included hand-coded routines for such things as sometimes standing still (giving the impression of “thinking about the next action”) or occasionally misjudging a jump can be very effective in giving the appearance of human playing style. Another way of providing human-like behavior is to explicitly **mimic** humans by learning from play traces. Ortega et al. describe a method for creating Mario-playing agents in the style of particular human players [511]: evolve neural networks where the fitness function is based on whether the agent would perform the same action as the human, when faced with the same situation. This was shown to generate more human-like behavior than evolving the same neural network architecture with a more straightforward fitness function. In a similar effort to create human-like Mario AI players Munoz et al. [469] used both play traces and information about the player’s eyes position on the screen (obtained via gaze tracking) as inputs of an ANN, which was trained to approximate which keyboard action is to be performed at each game step. Their results yield a high prediction accuracy of player actions and show promise towards the development of more human-like Mario controllers based on information beyond gameplay data.

### 3.4.3.3 The ALE Framework

The Arcade Learning Environment (ALE) is an environment for general game-playing research based on an emulation of the classic video game console *Atari 2600* [40]. (While the environment can technically accommodate other emulators as well, the Atari 2600 emulator is the one that has been used in practice, to the point that the ALE framework is sometimes simply referred to as “Atari”.) The Atari 2600 is a console from 1976 with 128 bytes of RAM, maximum 32 kilobytes of ROM per game and no screen buffer, posing severe limitations on the type of games that could be implemented on the system [466]. ALE provides an interface for agents to control games via the standard joystick input, but does not provide any processed version of the internal state; instead, it provides the  $160 \times 210$  pixel screen output to the agent, which will need to parse this visual information somehow. There is a forward model, but it is relatively slow and generally not used.

Some of the early work using ALE used neuroevolution; in particular a study compared several neuroevolution algorithms on 61 Atari games [251]. They found that they could use the popular neuroevolution algorithm NEAT to evolve decent-quality players for individual games, provided that these algorithms were given positions of in-game objects as recognized by a computer vision algorithm. The HyperNEAT algorithm, an indirect encoding neuroevolution algorithm that can create arbitrarily large networks, was able to learn agents that could play based on the raw pixel inputs, even surpassing human performance in three of the tested games. In that paper, the neuroevolution approaches generally performed much better than the classic reinforcement learning methods tried.

Later, ALE was used in Google DeepMind’s research on deep Q-learning, which was reported in a *Nature* paper in 2015 [464]. As detailed in Chapter 2, the study

showed that by training a deep neural network (five layers, where the first two are convolutional) with Q-learning augmented with experience replay, human-level playing performance could be reached in 29 out of 49 tested Atari games. That research spurred a flurry of experiments in trying to improve the core deep reinforcement formula presented in that paper.

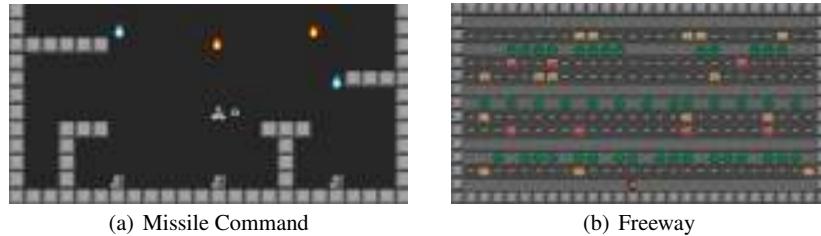
It is worth noting that in almost all of the ALE work focuses on learning neural networks (or occasionally other agent representations) for individual games. That is, the network architecture and input representation is the same across all games, but the parameters (network weights) are learned for a single game and can only play that game. This seems to be at odds with the idea of general game playing, i.e., that you could learn agents that play not a single game, but any game you give them. It could also be noted that ALE itself is better suited for research into playing individual games than for research on general game playing, as there is only a limited number of Atari 2600 games, and it is highly non-trivial to create new games for this platform. This makes it possible to tune architectures and even agents to individual games.

#### 3.4.3.4 General Video Game AI

The General Video Game AI (Gvgai) competition is a game-based AI competition that has been running since 2014 [528]. It was designed partly as a response to a trend seen in many of the existing game-based AI competitions, e.g., those organized at the CIG and AIIDE conferences, that submissions were getting increasingly game-specific by incorporating more and more domain knowledge. A central idea of Gvgai is therefore that submissions to the competition are tested on *unseen* games, i.e., games that have not been released to the competitors before and which are therefore impossible to tailor the submissions to. At the time of writing, the Gvgai repository includes around 100 games, with ten more games being added for every competition event.

To simplify the development of games in Gvgai, a language called the Video Game Description Language (VGDL) was developed [181, 597]. This language allows for concise specification of games in a Python-like syntax; a typical game description is 20-30 lines, with levels specified in separate files. Given the underlying assumptions of 2D movement and graphical logics, most of the games in Gvgai corpus are remakes of (or inspired by) classic arcade games such as *Frogger* (Sega, 1981) (see Fig. 3.8(b)), *Boulder Dash* (First Star Software, 1984) or *Space Invaders* (Taito, 1978) (see Fig. 3.8(a)), but some are versions of modern indie games such as *A Good Snow Man Is Hard to Build* (Hazelden and Davis, 2015).

The original track of the Gvgai competition, for which the most results are available, is the single-player planning track. Here, agents are given a fast forward model of the game, and 40 milliseconds to use it to plan for the next action. Given these conditions, it stands to reason that planning algorithms of various kinds would rule the day. Most of the top performers in this track have been based on variations on MCTS or MCTS-like algorithms, such as the Open Loop Expectimax



**Fig. 3.8** Two example games that have been used in the GVGAI competition.

Tree Search algorithm [528] or MCTS with options search [161]. One surprisingly high-performing agent uses Iterative Width search, where the core idea is to build a propositional database of all the facts and then use this to prune a breadth-first search algorithm to only explore those branches which provide a specified amount of novelty, as measured by the size of the smallest set of facts seen for the first time [389]. Agents based on evolutionary planning also perform well, but not as well as those based on stochastic tree search or tree search with novelty pruning.

While some agents are better than others overall, there is clear non-transitivity in the rankings, in the sense that the best algorithms for one particular game may not be the best for another game—in fact, there seem to be patterns where families of algorithms perform better on families of games [213, 59]. Given these patterns, a natural idea for achieving higher playing performance is to use **hyper-heuristics** or **algorithm selection** to select at runtime which algorithm to use for which game, an approach which has seen some success so far [453].

Two other GVGAI tracks are related to gameplay, namely the two-player planning track and the learning track. The two-player planning track resembles the single-player planning track but features a number of two-player games, some of which are cooperative and some competitive. At the time of writing, the best agents in this track are slightly modified versions of single-player track agents that make naive assumptions about the behavior of the other player [216]; it is expected that agents with more sophisticated player models will eventually do better. The learning track, by contrast, features single-player games and provides players with time to learn a policy but does *not* provide them with a forward model. It is expected that algorithms such as deep reinforcement learning and neuroevolution will do well here, but there are as yet no results; it is conceivable that algorithms that learn a forward model and then perform tree search will dominate.

#### 3.4.3.5 Other Environments

In addition to ALE and GVGAI, there are several other environments that can be used for AI experimentation with arcade-style games. The *Retro Learning Environment* is a learning environment similar in concept to ALE, but instead based on an

emulation of the Super Nintendo console [45]. A more general, and less focused, system is *OpenAI Universe*,<sup>7</sup> which acts as a unified interface to a large number of different games, ranging from simple arcade games to complex modern adventure games.

### 3.4.4 Strategy Games

Strategy games, particularly **computer strategy games**, are games where the player controls multiple characters or units, and the objective of the game is to prevail in some sort of conquest or conflict. Usually, but not always, the narrative and graphics reflect a military conflict, where units may be e.g., knights, tanks or battleships. The perhaps most important distinction within strategy games is between **turn-based** and **real-time** strategy games, where the former leave plenty of time for the player to decide which actions to take each time, and the latter impose a time pressure. Well-known turn-based strategy games include epic strategy games such as the *Civilization* (MicroProse, 1991) and the *XCOM* (MicroProse, 1994) series, as well as shorter games such as *Hero Academy* (Robot Entertainment, 2012). Prominent real-time strategy games include *StarCraft I* (Blizzard Entertainment, 1998) and II (Blizzard Entertainment, 2010), the *Age of Empires* (Microsoft Studios, 1997–2016) series and the *Command and Conquer* (Electronic Arts, 1995–2013) series. Another distinction is between single-player games that focus on exploration, such as the *Civilization* games, and multi-player competitive games such as *StarCraft* (Blizzard Entertainment, 1998–2015). Most, but not all, strategy games feature hidden information.

The cognitive challenge in strategy games is to lay and execute complex plans involving multiple units. This challenge is in general significantly harder than the planning challenge in classical board games such as Chess mainly because multiple units must be moved at every turn; the number of units a player controls can easily exceed the limits of short-term memory. The planning horizon can be extremely long, where for example in *Civilization V* (2K Games, 2010) decisions you make regarding the building of individual cities will affect gameplay for several hundred turns. The order in which units are moved can significantly affect the outcome of a move, in particular because a single action might reveal new information during a move, leading to a prioritization challenge. In addition, there is the challenge of predicting the moves of one or several adversaries, who frequently have multiple units. For real-time strategy games, there are additional perceptual and motoric challenges related to the speed of the game. This cognitive complexity is mirrored in the computational complexity for agents playing these games—as discussed in Section 3.2.1.4, the branching factor for a strategy game can easily reach millions or more.

---

<sup>7</sup> <https://universe.openai.com/>

The massive search spaces and large branching factors of strategy games pose serious problems for most search algorithms, as just searching a single turn forward might already be infeasible. One way of handling this is to decompose the problem, so each unit acts on its own; this creates one search problem for each unit, with the branching factor equivalent to the branching factor of the individual unit. This has the advantage of being tractable and the disadvantage of preventing coordinated action among units. Nevertheless, heuristic approaches where units are treated separately are used in the built-in AI in many strategy games. Not coincidentally the built-in AI of many strategy games is generally considered inadequate.

In research on playing strategy games, some solutions to this involve cleverly sub-sampling the space of turns so that standard search algorithms can be used. An example of such an approach is an MCTS variant based on decomposition through Naive Sampling [503]. Another approach is non-linear Monte Carlo, which was applied to *Civilization II* (MicroProse, 1996) with very promising results [65]. The basic idea here is to sample the space of turns (where each turn consists of actions for all units) randomly, and get an estimate for the value of each turn by performing a rollout (take random actions) until a specified point. Based on these estimates, a neural network was trained to predict the value of turns; regression can then be used to search for the turn with the highest predicted value.

But planning does not need to be based on tree search. Justesen et al. applied online evolutionary planning to *Hero Academy* (Robot Entertainment, 2012), a two-player competitive strategy game with perfect information and a relatively low number of moves per turn (five in the standard setting) [309]. Each chromosome consisted of the actions to take during a single turn, with the fitness function being the material difference at the end of the turn. It was found that this approach vastly outperformed MCTS (and other tree search algorithms) despite the shallow search depth of a single turn, likely because the branching factor made it impossible for the tree search algorithms to sufficiently explore even the space of this single turn.

Evolution has also been used to create agents that play strategy games. For example, NEAT-based macro-management controllers were trained for the strategy game *Globulation 2* (2009). In that study, however, the NEAT controller does not aim to win but rather play for the experience; in particular, it is evolved to take macro-actions (e.g., build planning, battle planning) in order to provide a balanced game for all players [499]. AI approaches based on artificial evolution that also have been used as playtesting mechanisms for a number of other strategy games [588, 297].

#### 3.4.4.1 StarCraft

The original *StarCraft*, released in 1998 by Blizzard Entertainment, is still widely played competitively, a testament to its strong game design and in particular its unusual depth of challenge (see Fig. 3.10). It is usually played with the *Brood War* expansion, and referred to as *SC:BW*. The existence of *Brood War API (BWAPI)*,<sup>8</sup> an

---

<sup>8</sup> <https://github.com/bwapi/bwapi>

interface for playing the game with artificial agents, has enabled a thriving community of researchers and hobbyists working an AI for *StarCraft* (Blizzard Entertainment, 1998). Several competitions are held annually based on *SC:BW* and *BWAPI*,<sup>9</sup> including competitions at the IEEE CIG and AIIDE conferences [87]. *TorchCraft* is an environment built on top of *SC:BW* and *BWAPI* to facilitate machine learning, especially deep learning, research using *StarCraft* [681]. In parallel, a similar API was recently released for interfacing AI agents with the follow-up game *Starcraft II* (Blizzard Entertainment, 2010), which is mechanically and conceptually very similar but with numerous technical differences. Given the existing API and competitions, almost all existing research has been done using *SC:BW*.

As *SC:BW* is such a complex game and the challenge of playing it well is so immense, most research focuses on only part of the problem, most commonly through playing it at some level of abstraction. It is common to divide the different levels of decision-making in *SC:BW* (and similar real-time strategy games) into three levels, depending on the time scale: Strategy, Tactics and Micro-Management (see Fig. 3.9). So far, no agents have been developed that can play a complete game at the level of even an intermediate human player; however, there has been significant progress on several sub-problems of the very formidable problem of playing this game.

For a fuller overview of research on AI for playing *SC:BW*, the reader is referred to a recent survey [504]. Below, we will exemplify some of the research done in this space, with no pretense of complete coverage.

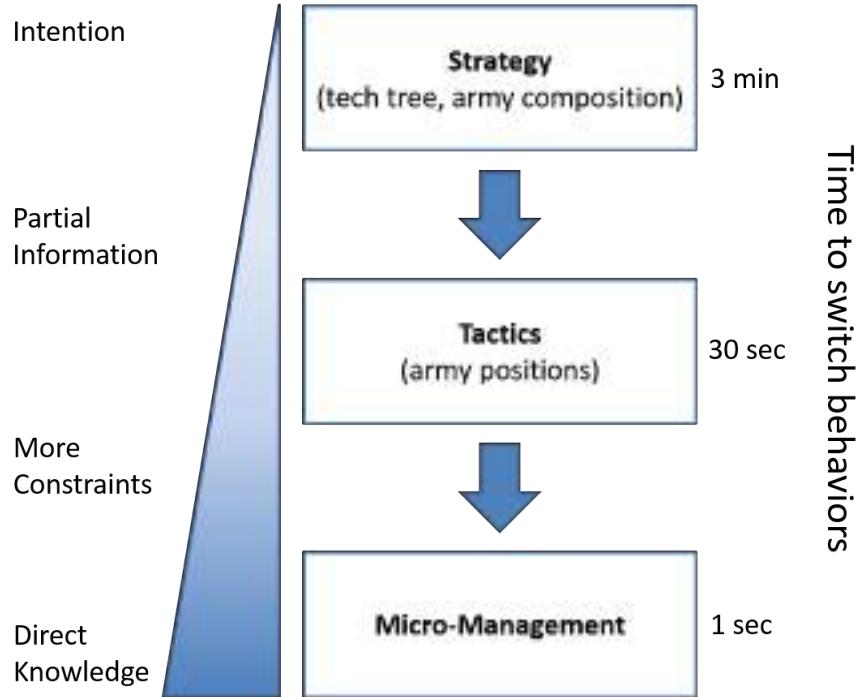
On the **micro** level AI plays out over the timescale of usually less than a minute, where time between taking actions is typically on the order of a second. When focusing on this most low-level form of *SC:BW* battle, one need not consider base building, research, fog of war, exploration and many other aspects of the full *SC:BW* game. Usually, two factions face off, each with a set of a few or a few dozen units. The goal is to destroy the opponent’s units. This game mode can be played in the actual game, which does not allow for significant speed-up and does not provide a forward model, or in the simulator SparCraft [123], which does provide a forward model. (There is also a Java version of SparCraft, called *JarCraft* [311].)

In the model-free scenario, agents must be based on either hand-crafted policies or policies learned through reinforcement learning or supervised learning, without the luxury of a forward model. Hand-crafted policies can be implemented e.g., based on potential fields, where different units are attracted to or repelled by other units in order to create effective combat patterns [242], or on fuzzy logic [541]. When it comes to machine learning methods for model-free scenarios, standard [622] and deep reinforcement learning [729] have been used with some effect to learn policies. Typically, the problem is decomposed so that a single Q-function is learned that is then applied to each unit separately [729].

Using the SparCraft simulator, we can do more because of the availability of a forward model. Churchill and Buro developed a simple approach to dealing with the excessive branching factor called Portfolio Greedy Search [123]. The core idea is

---

<sup>9</sup> <http://www.starcraftai.com/>



**Fig. 3.9** Three different levels of decision making in *StarCraft* (Blizzard Entertainment, 1998). The width of the triangle represents amount of information whereas its colored gradient illustrates the degree of partial observability. For instance, at the highest strategic level both the level of observability and available information to the player are relatively low. On the other end of the triangle, at the lowest level of micro-management the player must consider the type, position, and other dynamic properties of each of the units she controls; that information is mostly observable. For reference, a full game of *StarCraft* (Blizzard Entertainment, 1998) often takes around 20 minutes, though there is considerable variation. The image is reproduced with permission from Gabriel Synnaeve.

that instead of selecting between actions for each unit, a small number (a portfolio) of simple heuristics called scripts are used, and a greedy search algorithm is used to assign one of these scripts to each unit. This approach prunes the branching drastically, but limits the space of discoverable policies to those that can be described as combinations of the scripts. Subsequently, it was shown that the portfolio of scripts idea can be combined with MCTS with good results [311]. Even better results can be obtained by doing portfolio selection through evolutionary planning [745]; it is likely that these ideas generalize to many strategy games and other games with high branching factors due to their controlling of many units simultaneously.

Moving to the other end of the micro-management-tactics-strategy continuum, large-scale **strategy** adaptation remains a very hard problem. Existing *SC:BW* bots are rarely able to implement multiple strategies, let alone adapt their strategy based on how the game progresses. In order to do this, it is necessary to create a model of



**Fig. 3.10** A screenshot from *StarCraft: Brood War* (Blizzard Entertainment, 1998) displaying a number of different units available in the game. At the time of writing, playing well this real-time strategy game is considered one of the next grand challenges for AI research. Image obtained from Wikipedia (fair use).

what the opponent is trying to do based on limited evidence. Here, pioneering work by Weber and Mateas focused on mining logs of *SC:BW* matches to predict what strategy will be taken by a player from early-game actions [750].

A few more ambitious attempts have been made to create complete agents that can handle strategy, tactics and micro-management in a principled fashion. For example, Synnaeve and Bessière built an agent based on Bayesian programming that is able to perform reasonably well [680].

### 3.4.5 Racing Games

**Racing games** are games where the player is tasked with controlling some kind of vehicle or character so as to reach a goal in the shortest possible time, or as to traverse as far as possible along a track in a given time. Usually the game employs a first-person perspective, or a vantage point from just behind the player-controlled vehicle. The vast majority of racing games take a continuous input signal as a steering input, similar to a steering wheel. Some games, such as those in the *Forza Mo-*



**Fig. 3.11** A screenshot from TORCS which has been predominately used in the Simulated Car Racing Championship. Image obtained from <https://sourceforge.net/projects/torcs/> (fair use).

*torsport* (Microsoft Studios, 2005–2016) or *Real Racing* (Firemint and EA Games, 2009–2013) series, allow for complex input including gear stick, clutch and hand-brake, whereas more arcade-focused games such as those in the *Need for Speed* (Electronic Arts, 1994–2015) series typically have a simpler set of inputs and thus lower branching factor. Racing games such as those in the *WipeOut* (Sony Computer Entertainment Europe, 1995–2012) and *Mario Kart* (Nintendo, 1992–2017) series introduce additional elements, such as weapons that can be used to temporarily incapacitate competitors' vehicles.

While the cognitive challenges in playing racing games may appear simple, most racing games actually require multiple simultaneous tasks to be executed and have significant skill depth. At the most basic level, the agent needs to control for the position of the vehicle and adjust the acceleration or braking, using fine-tuned continuous input, so as to traverse the track as fast as possible. Doing this optimally requires at least short-term planning, one or two turns (of the track) forward. If there are resources to be managed in the game, such as fuel, damage or speed boosts, this requires longer-term planning. When other vehicles are present on the track, there is an adversarial planning aspect added, in trying to manage or block overtaking; this planning is often done in the presence of hidden information (position and resources of other vehicles on different parts of the track) and under considerable time pressure, and benefits from models of the adversarial drivers.

One relatively early commercial game application that stands out is the AI for *Forza Motorsport* (Microsoft Studios, 2005), which was marketed under the name *Drivatar* [259]. The Drivatar agents are built on a form of supervised lazy learning. To train the agents, humans drive a number of racing tracks, which are composed of a number of segments; all tracks in the game need to be composed of segments

drawn from the same “alphabet”. During driving, the agent selects the driving commands that most closely approximate the racing line taken by the players on the relevant segment. This approach was successful in realizing personalized driving agents, i.e., agents that could drive new tracks in the style of human players they had been trained on, but posed restrictions on the design of the tracks.

There are various approaches to training agents to drive without supervision, through reinforcement learning. A sequence of papers has shown how neuroevolution can be used to train agents that drive a single track in the absence of other cars as well as a good human driver [707], how incremental evolution can be used to train agents with sufficiently general driving skills to drive unseen tracks [709], and how competitive co-evolution can be used to adversarially train agents to drive more or less aggressively in the presence of other cars [708]. In all these experiments, the weights of a relatively small fixed-topology network were trained with an evolution strategy. The inputs to the network were the speed of the car and a handful of rangefinder sensors that returned the distance to the edges of the track, or other cars. The low dimensionality of the resulting network enabled high-performing networks to be found relatively easily.

The *Simulated Car Racing Championship*, which has run annually since 2007, is partly based on this work, and uses a similar sensor model. The first year, the competition was based on simple 2D racing game, and the winner of the competition was a controller based on fuzzy logic [710]. In 2008, the competition software was rebuilt around TORCS, a 3D racing game with a reasonably sophisticated physics model [393] (see Fig. 3.11). In the following years, a large number of competitors submitted agents based on various different architectures to the competition, including evolutionary computation, temporal difference learning, supervised learning and simple hand-coded rule-based systems [393]. A general trend has been observed over the course of the competition that the winning agents incorporate more and more domain knowledge in the form of hand-coded mechanisms, with learning algorithms generally only used for tuning parameters of these mechanisms. The best agents, such as *COBOSTAR* [90] or *Mr. Racer* [543] generally drive as well as or better than a good human driver when driving alone on a track, but still struggle with overtaking and other forms of adversarial driving.

As discussed above, the Simulated Car Racing Championship provides information in a form that is relatively easy to map to driving commands, making the learning of at least basic driving strategy (but not fine-tuning) relatively easy. However, some authors have attempted learning to drive from raw pixel data. Early work on this topic includes that by Floreano et al., who evolved a neural network with a movable “retina” to drive in a simple simulated environment. The output of the neural network included both driving commands and commands for how to move the retina, and only the relatively few pixels in the retina were used as inputs to the network [206]. Later, Koutnik et al. managed to evolve controllers that used higher-dimensional input by evolving the networks in compressed weight space; essentially, the parameters of a JPEG encoding of the network connections was evolved, allowing evolutionary search to work effectively in the space of large neural net-

works [353]. Supervised learning of deep networks has also been applied to visual driving, yielding high-performing TORCS drivers that learn from examples [795].

The examples above do not make use of a forward model of any kind. However, car dynamics are relatively simple to model, and it is easy to create a fast approximate model for racing games. Given such a model, standard tree search algorithms can easily be applied to car control. For example, Fischer et al. showed that MCTS coupled with a simple forward model can produce decent performance in TORCS [203].

### 3.4.6 Shooters and Other First-Person Games

**First-person shooters** constitute an important genre of video games ever since the success of *DOOM* (GT Interactive, 1993) and *Wolfenstein 3D* (Apogee Software and FormGen, 1992) in the early 1990s. While a basic tenet of an FPS would seem to be that the world is observed through a first-person point of view, there are games that are generally recognized as FPSes, such as the *Gears of War* (Microsoft Studios, 2006–2016) series, which have the camera positioned slightly behind and/or above the player. Similarly, the word “shooter” signifies that the games revolve around shooting projectiles with some kind of weapon. On that basis, a game such as *Portal* (Electronic Arts, 2007) can be seen as an FPS though it is debatable whether the player implement is actually a weapon.

Shooters are often seen as fast-paced games where speed of perception and reaction is crucial, and this is true to an extent, although the speed of gameplay varies between different shooters. Obviously, quick reactions are in general not a problem for a computer program, meaning that an AI player has a certain advantage over a human by default. But there are other cognitive challenges as well, including orientation and movement in a complex three-dimensional environment, predicting actions and locations of multiple adversaries, and in some game modes also team-based collaboration. If visual inputs are used, there is the added challenge of extracting relevant information from pixels.

There has been some early work on optimizing parameters for existing agents in order to improve their efficiency [127], but extensive work on AI for FPS games was spurred by two competitions: first, the **2K BotPrize** and more recently **VizDoom**.

#### 3.4.6.1 Unreal Tournament 2004 and the 2K BotPrize

*Unreal Tournament 2004* (UT2k4) (Epic Games, 2004) is an FPS which was released in 2004, with what was at the time state-of-the-art graphics and gameplay. While the game itself has not been open sourced, a team based at the Charles University in Prague created *Pogamut*, a Java-based API that allows for simple control of the game [222]. Pogamut supplies the agent with an object-based information interface, which the agent can query about the locations of objects and characters, and



**Fig. 3.12** The first 2K BotPrize competition held in Perth, Australia, on 17 December 2008, as part of the 2008 IEEE Symposium on Computational Intelligence and Games.

also provides convenience functions for executing actions such as firing a projectile towards a specific point.

Some work using UT2k4 tries to achieve high-performing agents for one or several in-game tasks, using techniques such as neuroevolution. For example, van Hoorn et al. subdivided the task of playing UT2k4 into three sub-tasks: shooting, exploring and path-following [734]. Using an earlier approach [698], which combines neuroevolution with the subsumption architecture of Rodney Brooks [70], they then evolved neural networks for each of these tasks in succession. The resulting agent was able to play some game scenarios relatively effectively.

However, the main use of the UT2k4 benchmark has been in the **2K BotPrize** (see Fig. 3.12). This competition, which ran from 2008 to 2014, stands out among game-based AI competitions for not focusing on playing for performance, but rather for playing for experience. Specifically, it was a form of Turing test, where submitted agents were judged not by how well they survived firefights with other agents, but by whether they could fool human judges (who in later configurations of the competition also participated in the game) that they were humans [262, 263, 264].

The winners of the final 2K BotPrize in 2014 were two teams whose bots managed to convince more than half of the human judges that they (the bots) were human. The first winning team, UT<sup>~</sup>2, from the University of Texas at Austin, is primarily based on neuroevolution through multiobjective evolution [603]. It consists of a number of separate controllers, where most of these are based on neural networks; at each frame, it cycles through all of these controllers, and uses a set of priorities to decide the outputs of which controller will command various aspects of the agent. In addition to neural networks, some controllers are built on different principles, in particular the Human Retrace Controller, which uses traces of human players to help the agent navigate out of stuck positions. The second winner, *MirrorBot* by Mihai Polceanu, is built around the idea of observing other players in the game and mirroring their behavior [535].

### 3.4.6.2 Raw Screen Inputs and the Visual Doom AI Challenge

The **VizDoom** framework [333] is build around a version of the classic *DOOM* (GT Interactive, 1993) FPS game that allows researchers to develop AI bots that play the game using only the screen buffer. VizDoom was developed as an AI testbed by a team of researchers at the Institute of Computing Science, Poznan University of Technology (see Fig. 3.13). The framework includes several tasks of varying complexity, from health pack collection and maze navigation to all-out deathmatch. An annual competition based on VizDoom is held at the IEEE CIG conference since 2016, and the framework is also included in *OpenAI Gym*,<sup>10</sup> a collection of games which can be used for AI research.

Most of the published work on VizDoom has been based on deep reinforcement learning with convolutional neural networks, given that method's proven strength in learning to act based on raw pixel inputs. For example, *Arnold*, a well-performing agent in the first VizDoom competition, is based on deep reinforcement learning of two different networks, one for exploration and one for fighting [115].

But it is also possible to use evolutionary computation to train neural network controllers. As the very large input size requires huge networks, which do not work well with evolutionary optimization in general, it is necessary to compress the information somehow. This can be done by using an autoencoder trained on the visual stream as the game is played; the activations of the bottleneck layer of the autoencoder can then be used as inputs to a neural network that decides about the actions, and the weights of the neural network can be evolved [12]. Previous attempts at evolving controllers acting on visual input in the related game *Quake* (GT Interactive, 1996) have met with only limited success [519].

### 3.4.7 Serious Games

The genre of **serious games**, or **games with a purpose** beyond entertainment, has become a focus domain of recent studies in game AI. One could argue that most existing games are serious by nature as they incorporate some form of learning for the player during play. Games such as *Minecraft* (Mojang, 2011), for example, were not designed with a particular learning objective in mind; nevertheless they have been used broadly in classrooms for science education. Further, one could argue that serious games do not have a particular genre of their own; games may have a purpose regardless of the genre they were designed on. Strictly speaking the design of serious games involves a particular set of learning objectives. Learning objectives may be educational objectives such as those considered in STEM education—a popular example of such a game is the *Dragonbox* (WeWantToKnow, 2011) series that teaches primary school students equation solving skills, and basic addition and subtraction skills. (A serious academic effort on game-based STEM

<sup>10</sup> <https://gym.openai.com/>



**Fig. 3.13** A screenshot from the VizDoom framework which—at the moment of writing—is used in the Visual Doom AI Challenge. The framework is giving access to the depth of the level (enabling 3D vision). Image obtained from <http://vizdoom.cs.put.edu.pl/> with permission.

education is the narrative-centered *Crystal Island* game series for effective science learning [577, 584].) The learning objective can, instead, be the training of social skills such as conflict resolution and social inclusion through games; *Village Voices* [336] (see Fig. 3.14(a)), *My Dream Theater* [100] (see Fig. 3.14(b)), and *Prom Week* [447] are examples of such soft skill training games. Alternatively, the aim could be that war veterans suffering from post-traumatic stress disorder are trained to cope with their cognitive-behavioral manifestations when faced with in-game stressors in games such as *StartleMart* [272, 270] and *Virtual Iraq* [227]. The learning objective could also be that of soliciting collective intelligence for scientists. A number of scientific games have recently led to the discovery of new knowledge via crowd-playing (or else human computation); arguably one of the most popular scientific discovery games is *Foldit* [138] through which players collectively discovered a novel algorithm for protein folding.

The cognitive and emotional skills required to play a serious game largely depend on the game and the underlying learning objectives. A game about math would normally require computation and problem solving skills. A game about stress inoculation and exposure therapy would instead require cognitive-behavioral coping mechanisms, metacognition and self-control of cognitive appraisal. The breadth of



(a) *Village Voices*. Screenshot image adapted from [336].

(b) *My Dream Theater*. Screenshot image adapted from [99].

**Fig. 3.14** The *Village Voices* and the *My Dream Theater* games for conflict resolution. *Village Voices* realizes experiential learning of conflict in social, multi-player settings, whereas *My Dream Theater* offers a single-player conflict management experience. In the first game the AI takes the role of modeling conflict and generating appropriate quests for the players, whereas—more relevant to the aims of this chapter—in *My Dream Theater* AI takes the role of controlling expressive agent (NPC) behaviors.

cognitive and emotional skills required from players is as wide as the number of different learning objectives a serious game can integrate into its design.

Many serious games have NPCs and AI can help in making those NPCs believable, human-like, social and expressive. AI in serious games is generally useful for modeling NPC behavior and playing the game as an NPC but not for winning; rather for the experience of play. Whether the game is for education, health or simulation purposes NPC agents need to act believably and emotively in order to empower learning or boost the engagement level of the game. Years of active research have been dedicated on this task within the fields of affective computing and virtual agents. The usual approach followed is the construction of top-down (ad-hoc designed) agent architectures that represent various cognitive, social, emotive and behavioral abilities. The focus has traditionally been on both the modeling of the agents behavior but also on its appropriate expression under particular contexts. A popular way of constructing a computational model of agent behavior is to base it on a theoretical cognitive model such as the OCC model [512, 183, 16, 189, 237], which attempts to effect human-like decision making, appraisal and coping mechanisms dependent on a set of perceived stimuli. For the interested reader, Marsella et al. [428] cover the most popular computational models of emotion for agents in a thorough manner.

Expressive and believable conversational agents such as *Greta* [534] and *Rea* [105] or virtual humans [674] that embody affective manifestations can be considered in the design of serious games. The use of such character models has been dominant in the domains of intelligent tutoring systems [131], embodied conversational agents [104, 16], and affective agents [238] for educational and health purposes. Notable examples of such agent architecture systems include the work of Lester's

group in the *Crystal Island* game series [578, 577, 584], the expressive agents of *Prom Week* [447] and *Façade* [441], the agents of the *World of Minds* [190] game, and the FAtiMA [168] enabled agents of *My Dream Theater* [100].

### 3.4.8 Interactive Fiction

While there are several variations, games within the **interactive fiction** genre normally contain a fantasy world consisting of smaller areas such as rooms; however, a simulated environment is not a necessity. Importantly, players need to use text commands to play the game. The player can normally interact with the objects and available game characters, collect objects and store them in her inventory, and solve various puzzles. Games of this genre are also named *text-based adventure* games or often associated with text-based *role-playing games*. Popular examples include the games of the *Zork* series (Infocom, 1979–1982) and *Façade* [441].

In this game genre, AI can play the role of understanding text as coming from players in a natural language format. In other words, the game AI can feature **natural language processing** (NLP) for playing the game as the player's companion or as an opponent. Further NLP can be used as an input for the generation of a dialog, a text or a story in an interactive fashion with the player. It is normally the case that text-based input is used to drive a story (interactive narrative) which is often communicated via embodied conversational agents and is represented through the lens of a virtual camera. Similarly to traditional cinematography, both the camera position and the communicated narrative contribute to the experience of the viewer. Opposed to traditional cinematography, however (but similarly to interactive drama), the story in games can be influenced by the player herself. It goes without saying that research in text-based games is naturally interwoven with research in believable conversational agents (as covered in the previous section), computational and **interactive narrative** [693, 441, 562, 792] and **virtual cinematography** [252, 193, 300, 84, 15, 578]. The discussion on the interplay between interactive narrative and virtual cinematography is expanded in Chapter 4 and in particular in the section dedicated to narrative generation.

Work on text-based AI in games starts from the early language-based interaction with Eliza [751] and the Z-Machine used by text adventure games such as *Zork I* (Infocom, 1980), to *Façade* [438, 441] and to recent word2vec [459] approaches (e.g., its TensorFlow implementation [2]) for playing Q & A games [340] and text-based adventure games [352]. It is important to note that beyond the use of AI to understand natural language we can use AI to play text-based games. A notable recent example of an agent that manages to handle both tasks is the one developed by Kostka et al. [352], named *Golovin*. The *Golovin* agent uses related corpora such as fantasy books to create language models (via word2vec [459]) that are appropriate to this game domain. To play the game the agent uses five types of command generators: battle mode, gathering items, inventory commands, general actions and movement. *Golovin* is validated on 50 interactive fiction games demonstrating com-

parable performance to the current state of the art. Another example is the agent of Narasimhan et al. [475] that plays *multi-user dungeon* games, a form of multi-player or collaborative interactive fiction. Their agent converts text representations to state representations using Long Short-Term Memory (LSTM) networks. These representations feed a deep Q network which learns approximate evaluations for each action in a given game state [475]. Their approach significantly outperforms other baselines in terms of number of completed quests in small-, and even medium-sized, games. For the interested reader a dedicated annual competition on text-based adventure game AI was initiated in conjunction with the IEEE CIG conference in 2016.<sup>11</sup> Participants of the competition submit agents that play games for the Z-Machine.

Examples of useful development tools for text-based games include the *Inform* series of design systems and programming languages, which are inspired by the Z-machine and have led to the development of several text-based games and interactive fiction based on natural language. Notably, *Inform 7*<sup>12</sup> was used for the design of the *Mystery House Possessed* (Emily Short, 2005) game.

### 3.4.9 Other Games

The list of games AI can play is not limited to the genres covered above. While the genres we covered in more detail are, in our opinion, the most representative there are AI studies focusing on other game genres, a number of which we outline below.

A game type with an increasing interest is **casual games** due to their growing popularity and accessibility via mobile devices in recent years. Casual games are often simple and are designed as short episodes of play (levels) to allow flexibility with respect to gameplay time. This feature gives the player the ability to conclude an episode in a short period of time without needing to save the game. As a result, the player can engage on a single level for a few seconds, or play a number of levels throughout the day, or instead repeatedly play new levels up to hours of gameplay. The game skills required to play casual games depend on the genre of the casual game which can vary from puzzle such as *Bejeweled* (PopCap Games, 2001), *Angry Birds* (Chillingo, 2009), and *Cut the Rope* (Chillingo, 2010), to adventure such as *Dream Chronicles* (KatGames, 2007), to strategy such as *Diner Dash* (PlayFirst, 2004), to arcade such as *Plants vs. Zombies* (PopCap Games, 2009) and *Feeding Frenzy* (PopCap Games, 2004), to card and board games such as *Slingo Quest* (funkitron, Inc., 2006).

Notable academic efforts on casual games include the work of Isaksen et al. [288, 289] where an AI agent is built to test the difficulty of *Flappy Bird* (dot-GEARS, 2013) levels. The baseline AI player follows a simple pathfinding algorithm that performs well in completing the *Flappy Bird* levels. To imitate human

---

<sup>11</sup> <http://atkrye.github.io/IEEE-CIG-Text-Adventurer-Competition/>

<sup>12</sup> <http://inform7.com/>

play, however, the AI player features elements of human motor skills such as precision, reaction time, and actions per second. Another example on that line of work is the use of AI agents to test the generation of game levels in a variant of *Cut the Rope* (Chillingo, 2010). The AI agents featured in the *Ropossum* authoring tool both perform automatic playtesting and also optimize the playability of a level using first-order logic [614]. The level generation elements of *Ropossum* are further discussed in the next chapter. Another casual game that has recently attracted the interest of game AI research is *Angry Birds* (Chillingo, 2009). The game has an established AI competition [560], named the *Angry Birds AI Competition*,<sup>13</sup> that runs since 2012 mainly in conjunction with the International Joint Conference on Artificial Intelligence. AI approaches in *Angry Birds* (Chillingo, 2009) have so far focused mainly on planning and reasoning techniques. Examples include a qualitative spatial reasoning approach which evaluates level structural properties and game rules, and infers which of these are satisfied for each building block of the level [796]. The usefulness of each level building block (i.e., how good it is to hit it) is then computed based on these requirements. Other approaches model discrete knowledge about the current game state of *Angry Birds* (Chillingo, 2009) and then attempt to satisfy the constraints of the modeled world [92] based on extensions of answer set programming [69].

Beyond casual games the genre of **fighting games** has received a considerable amount of interest from both academic and industrial players. Fighting games require cognitive skills mostly related to kinesthetic control and spatial navigation but also related to reaction times and decision making, both of which need to be fast [349]. Popular approaches for fighting games include classic reinforcement learning—in particular, the SARSA algorithm for on-policy learning of Q values which are represented by linear and ANN function approximators—as applied by the Microsoft Research team in the *Tao Feng: Fist of the Lotus* (Microsoft Game Studios, 2003) game [235]. Reinforcement learning has also been applied with varying degrees of success for adaptive difficulty adjustment in fighting games [158, 561, 27]; i.e., AI that plays for the experience of the player. Evolutionary reinforcement learning variants have also been investigated for the task [437]. A notable effort in the fighting games AI research scene has been on the Java-based fighting game *FightingICE* provided by the *Fighting Game AI Competition*<sup>14</sup> [402] (see Fig. 3.15). The competition is organized by the Ritsumeikan University in Japan and has run since 2013 with the aim to derive the best possible fighting bot; i.e., AI that plays to win. Approaches using *FightingICE* vary from dynamic scripting [417], to k-nearest neighbor [760], to Monte Carlo tree search [790], to neuroevolution [357], among others. So far MCTS-based approaches appear to be advantageous on winning in fighting games.

The last game we will cover in this section is *Minecraft* (Mojang, 2011) for its unique properties as a testbed for game AI research. *Minecraft* (Mojang, 2011) is a **sandbox** game played in a 3D procedurally generated world that players can navi-

---

<sup>13</sup> <https://aibirds.org/>

<sup>14</sup> <http://www.ice.ci.ritsumei.ac.jp/~ftgaic/>



**Fig. 3.15** A screenshot from the Java-based *FightingICE* framework for fighting games.

gate through. The game features game mechanics that enable players to build constructions out of cubes (see Fig. 3.16) but it does not have a specific goal for the player to accomplish. Beyond exploration and building, players can also gather resources, craft objects and combat opponents. The game has sold more than 121 million copies across all platforms [51], making it the second best-selling video game of all time, only behind *Tetris* (Alexey Pajitnov and Vladimir Pokhilko, 1984).<sup>15</sup> The 3D open-world nature of *Minecraft* (Mojang, 2011) and the lack of specific goals provides players ultimate freedom to play and explore the world in dissimilar ways. The benefits of playing *Minecraft* (Mojang, 2011) appear to be many and some of them have already been reported by the educational research community [479]. For example, the blocks available in the game can be arranged to produce any object a player might think of, thereby fostering the player's creativity [479] and diagrammatic lateral thinking [774]. Further, the blocks' functionalities that can be combined and extended may result in new knowledge for the players that is acquired gradually. Moreover, the simple stylized voxel-based graphics of the game allow the player to concentrate on the gameplaying and exploration tasks within a simple, yet aesthetically pleasing, environment. Overall, the multitude of reasons that make *Minecraft* (Mojang, 2011) so appealing to millions of players are also the reasons that make the game a great testbed for AI and game AI research. In particular, the game offers an AI player an open world ready to be explored with numerous possibilities for open-ended play. Further, in-game tasks for an AI agent vary from

<sup>15</sup> [https://en.wikipedia.org/wiki/List\\_of\\_best-selling\\_video\\_games](https://en.wikipedia.org/wiki/List_of_best-selling_video_games)



**Fig. 3.16** A screenshot from *Minecraft* (Mojang, 2011) showcasing a city hall structure made by the MCFRArchitect Build Team. Image obtained from Wikipedia (fair use).

exploration and seeking treasure to making objects and building structures, alone or as a team of agents.

A notable and recent effort on the use of *Minecraft* (Mojang, 2011) for AI is *Project Malmo* [305] which is supported by Microsoft Research. Project Malmo is a Java-based AI experimentation platform built as a game mod of the original game and designed to support research within the areas of robotics, computer vision, machine learning, planning, and multi-agent systems and general game AI [305]. Note that the open-source platform is accessible via GitHub.<sup>16</sup> Early experiments with AI in *Project Malmo* include the application of deep neural networks for navigating the 3D mazes [465] and for fighting the opponents of the game [726]. Beyond *Project Malmo*, it is also worth noting that various other mods of the game have been used directly for teaching robotics [11]—including algorithms for maze navigation and planning—and teaching general AI methods [38].

### 3.5 Further Reading

The methods used for playing games have an extensive body of literature that is covered in detail both in Chapter 2 and here. The different game genres that we covered in this chapter contain a corresponding literature the interested reader could use as a starting point for further exploration.

<sup>16</sup> <https://github.com/Microsoft/malmo#getting-started>

## 3.6 Exercises

Prior to moving to the next chapter about generating content herein we provide a general exercise that allows you to apply any of the algorithms covered in Chapter 2 in a single domain. The purpose of the exercise is for you to get familiar with those methods before testing them in more complicated and complex domains and tasks.

As discussed above, the *Ms Pac-man vs Ghost Team* competition is a contest held at several AI conferences around the world in which AI controllers for Ms Pac-Man and the ghost team compete for the highest ranking. For this exercise, you will have to **develop a number of Ms Pac-Man AI players** to compete against the ghost team controllers included in the software package. This is a simulator entirely written in Java with a well-documented interface. While the selection of two to three different agents within a semester period has been shown to be a good educational practice we will leave the final number of Ms Pac-Man agents to be developed on you or your class instructor.

The website of the book contains code for the game and a number of different sample code classes in Java to get you started. All AI methods covered in Chapter 2 are applicable to the task of controlling Ms Pac-Man. You might find out, however, that some of them are more relevant and efficient than others. So which methods would work best? How do they compare in terms of performance? Which state representation should you use? Which utility function is the most appropriate? Do you think you can make the right implementation decisions so that your Ms Pac-Man plays at a professional level? How about at a world champion, or even at a superhuman level?

### 3.6.1 Why Ms Pac-Man?

Some of our readers might object to this choice of game and think that there should be more interesting games AI can play with. While *Ms Pac-Man* (Namco, 1982) is very old it is arguably a classic game that is still fun and challenging to play, as well as a simple testbed to start experimenting with the various AI methods and approaches introduced in this chapter and the previous one. *Ms Pac-Man* (Namco, 1982) is simple to understand and play but is not easy to master. This controversial element of play simplicity combined with problem complexity makes *Ms Pac-Man* (Namco, 1982) the ideal testbed for trying out different AI methodologies for controlling the main character, Ms Pac-Man. Another exciting feature of this game is its non-deterministic nature. Randomness not only augments the fun factor of the game but it also increases the challenge for any AI approach considered. As discussed, another argument for the selection of *Ms Pac-Man* (Namco, 1982) is that the game and its variants have been very well studied in the literature, tested through several game AI competitions, but also covered for years on AI (or game AI) courses in several universities across the globe. The reader is also referred to a recent survey paper

by Rohlfs hagen et al. [573], covering over 20 years of research in Pac-Man and a YouTube video<sup>17</sup> about the importance of Pac-Man in game AI research.

### 3.7 Summary

In this chapter we discussed the different roles AI can take and the different characteristics games and AI methods have, the various methods that are available for playing games and the different games it can play. In particular, AI can either play to win or play in order to create a particular experience for a human player or observer. The former goal involves maximizing a utility that maps to the game performance whereas the latter goal involves objectives beyond merely winning such as engagement, believability, balance and interestingness. AI as an actor can take the role of either a player character or a non-player character that exists in a game. The **characteristics** of games an AI method needs to consider when playing include the number of players, the level of stochasticity of the game, the amount of observability available, the action space and the branching factor, and the time granularity. Further, when we design an algorithm to play a game we also need to consider algorithmic aspects such as the state representation, the existence of a forward model, the training time available, and the number of games AI can play.

The above roles and characteristics were detailed in the first part of the chapter as they are important and relevant regardless of the AI method applied. When it comes to the **methods** covered in this chapter, we focused on tree-search, reinforcement learning, supervised learning, and hybrid approaches for playing games. In a sense, we tailored the methods outlined in Chapter 2 to the task of gameplaying. The chapter concluded with a detailed review of the studies and the relevant methodologies on a *game genre* basis. Specifically, we saw how AI can play board, card, arcade, strategy, racing, shooter, and serious games, interactive fiction, and a number of other game genres such as casual and fighting games.

---

<sup>17</sup> <https://www.youtube.com/watch?t=49&v=w5kFmdkrIuY>

## Chapter 4

# Generating Content

**Procedural content generation** (PCG) [616] is an area of game AI that has seen an explosive growth of interest. While games that incorporate some procedurally generated content have existed since the early 1980s—in particular, the dungeon crawler *Rogue* (Toy and Wichmann, 1980) and the space trading simulator *Elite* (Acornsoft, 1984) are early trailblazers—research interest in academia has really picked up within the second half of the last decade.

Simply put, PCG refers to methods for generating game content either autonomously or with only limited human input. Game **content** is that which is contained in a game: levels, maps, game rules, textures, stories, items, quests, music, weapons, vehicles, characters, etc. Typically, NPC behavior and the game engine itself are not thought of as content. Probably the most common current usage of PCG is for generating levels and terrains, but in the future we might see widespread generation of all kinds of content, possibly even complete games.

Seeing PCG from an AI perspective, content generation problems are AI problems where the solutions are content artifacts (e.g., levels) that fulfill certain constraints (e.g., being playable, having two exits) and/or maximize some metrics (e.g., length, difference in outcomes between different strategies). And as we will see, many AI methods discussed in Chapter 2 can be used for PCG purposes, including evolutionary algorithms and neural networks. But there are also a number of methods that are commonly used for PCG that are not typically thought of as AI; some of these will be presented in this chapter.

This chapter is devoted to methods for generating game content, as well as paradigms for how to incorporate them into games. We start with discussing why you want to use PCG at all—just like for playing games, there are some very different motivations for generating content. Then in Section 4.2 we present a general taxonomy for PCG methods and possible roles in games. Next, Section 4.3 summarizes the most important methods for the generation of content. Shifting attention to which roles PCG can take in games Section 4.4 discusses ways of involving designers and players in the generative process. Section 4.5 takes the perspective of content types, and presents examples of generating some common and uncommon types of game content. Finally, Section 4.6 discusses how to evaluate PCG methods.

## 4.1 Why Generate Content?

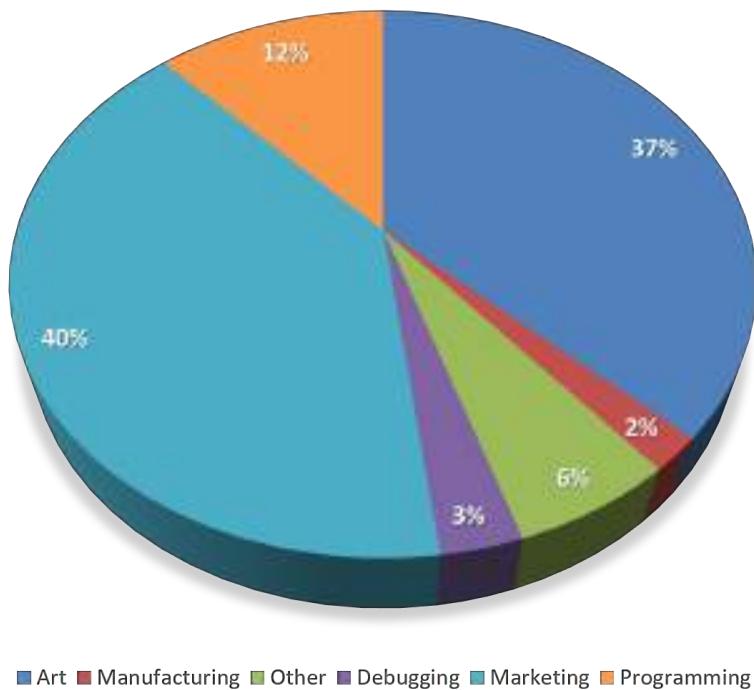
Perhaps the most obvious reason to generate content is that it could remove the need to have a human designer or artist creating that content. Humans are expensive and slow, and it seems we need more and more of them all the time. Ever since computer games were invented, the number of person-months that go into the development of a successful commercial game has increased more or less constantly.<sup>1</sup> It is now common for a game to be developed by hundreds of people over a period of several years. This leads to a situation where fewer games are profitable, and fewer developers can afford to develop a game, leading in turn to less risk-taking and less diversity in the games marketplace. Many of the costly employees necessary in this process are designers and artists rather than programmers. A game development company that could replace some of the artists and designers with algorithms would have a competitive advantage, as games could be produced **faster** and **cheaper** while preserving quality. (This argument was made forcefully by legendary game designer Will Wright in his talk “The Future of Content” at the 2005 Game Developers Conference, a talk which helped reinvigorate interest in procedural content generation.) Figure 4.1 illustrates a cost breakdown of an average AAA game and showcases the dominance of artwork and marketing in that process. Art, programming, and debugging constitute around 50% of the cost of an AAA game. Essentially, PCG can assist in the processes of art and content production, thus directly contributing to the reduction of around 40% of a game’s cost.

Of course, threatening to put them out of their jobs is not a great way to sell PCG to designers and artists. It is also true that at the current stage of technology, we are far from being able to replace all that a designer or artist can do. We could therefore turn the argument around: content generation, especially embedded in intelligent design tools, can augment the **creativity** of individual human creators. Humans, even those of the “creative” vein, tend to imitate each other and themselves. Algorithmic approaches might come up with radically different content than a human would create, through offering an unexpected but valid solution to a given content generation problem. Some evidence for this is already available in the literature [774]. This could make it possible for small teams without the resources of large companies, and even for hobbyists, to create content-rich games by freeing them from worrying about details and drudge work while retaining overall directorship of the games. PCG can then provide a way for *democratizing* game design by offering reliable and accessible ways for everyone to make better games in less time.

Both of these arguments assume that what we want to make is something like the games we have today. But PCG methods could also enable completely **new types of games**. To begin with, if we have software that can generate game content at the speed it is being “consumed” (played), there is in principle no reason why games need to end. For everyone who has ever been disappointed by their favorite game

---

<sup>1</sup> At least, this is true for “AAA” games, which are boxed games sold at full price worldwide. The recent rise of mobile games seems to have made single-person development feasible again, though average development costs are rising on that front too.

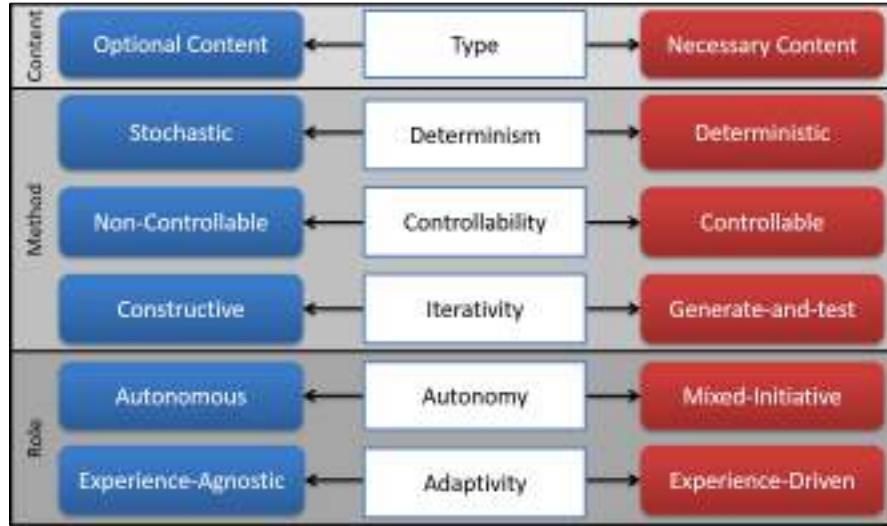


**Fig. 4.1** Average cost breakdown estimate of AAA game development. Data source: <http://monstervine.com/2013/06/chasing-an-industry-the-economics-of-videogames-turned-hollywood/>.

not having any more levels to clear, characters to meet, areas to explore, etc., this is an exciting prospect.

Even more excitingly, the newly generated content can be tailored to the tastes and needs of the player playing the game. By combining PCG with player modeling, for example through measuring and using neural networks to model the response of players to individual game elements, we can create **player-adaptive games** that seek to maximize the enjoyment of players (see the roles of PCG in Section 4.4). The same techniques could be used to maximize the learning effects of a serious game, or perhaps the addictiveness of a “casual” game.

Finally, a completely different but no less important reason for developing PCG methods is to **understand design and creativity**. Computer scientists are fond of saying that you do not really understand a process until you have implemented it in code (and the program runs). Creating software that can competently generate game content could help us understand the process by which we “manually” generate the content, and clarify the affordances and constraints of the design problem we are addressing. This is an iterative process, whereby better PCG methods can lead



**Fig. 4.2** An illustration of the PCG taxonomy discussed in Section 4.2.

to better understanding of the design process, which in turn can lead to better PCG algorithms enabling a co-creativity process between designers and algorithms [774].

## 4.2 Taxonomy

With the variety of content generation problems, methods and approaches that are available, it helps to have a structure that can highlight the differences and similarities between them. In the following, we introduce a revised version of the taxonomy of PCG that was originally presented by Togelius et al. [720]. It consists of a number of dimensions, where an individual method or solution should usually be thought of as lying somewhere on a continuum between the ends of that dimension. Beyond the taxonomy for the **content** types (Section 4.2.1) and the properties of the PCG **methods**, which we cover in Section 4.3, we provide an outline of the **roles** a PCG algorithm can take, which we cover extensively in Section 4.4.

### 4.2.1 Taxonomy for Content

We identify the type of the generated outcome as the sole dimension that relates to content in this taxonomy (see also Fig. 4.2).

### 4.2.1.1 Type: Necessary Versus Optional

PCG can be used to generate **necessary** game content that is required for the completion of a level, or it can be used to generate **optional** content that can be discarded or exchanged for other content. The main distinguishing feature between necessary and optional content is that necessary content should always be correct while this condition does not hold for optional content. Necessary needs to be consumed or passed as the player makes their way through the game, whereas optional content can be avoided or bypassed. An example of optional content is the generation of different types of weapons in first-person shooter games [240] or the auxiliary reward items in *Super Mario Bros* (Nintendo, 1985). Necessary content can be the main structure of the levels in *Super Mario Bros* (Nintendo, 1985), or the collection of certain items required to pass to the next level.

### 4.2.2 Taxonomy for Methods

PCG algorithms can be classified according to a number of properties such as their level of controllability, determinism and so on. This section outlines the three dimensions across which PCG methods can be classified. Figure 4.2 offers an illustration of the taxonomy that we discuss in this section.

#### 4.2.2.1 Determinism: Stochastic Versus Deterministic

Our first distinction with regards to PCG methods concerns the amount of randomness in content generation. The right amount of variation in outcomes between different runs of an algorithm with identical parameters is a design decision. **Stochasticity**<sup>2</sup> allows an algorithm (such as an evolutionary algorithm) to offer great variation, necessary for many PCG tasks, at the cost of controllability. While content diversity and expressivity are desired properties of generators, the effect of randomness on the final outcomes can only be observed and controlled after the fact. Completely **deterministic** PCG algorithms, on the other hand, can be seen as a form of data compression. A good example of deterministic PCG is the first-person shooter *.kkrieger* (.thepronukkt 2004), which manages to compress all of its textures, objects, music and levels together with its game engine in just 96 KB of storage space.

---

<sup>2</sup> Strictly speaking there is a distinction between a stochastic and a **non-deterministic** process in that the former has a defined random distribution whereas the latter does not. For the purposes of this book, however, we use the two terms interchangeably.

#### 4.2.2.2 Controllability: Controllable Versus Non-controllable

The generation of content by PCG can be controlled in different ways. The use of a random seed is one way to gain control over the generation space; another way is to use a set of parameters that control the content generation along a number of dimensions. Random seeds were used when generating the world in *Minecraft* (Mojang, 2011), which means the same world can be regenerated if the same seed is used [755]. A vector of content features was used in [617] to generate levels for *Infinite Mario Bros* (Persson, 2008) that satisfy a set of feature specifications.

#### 4.2.2.3 Iterativity: Constructive Versus Generate-and-Test

A final distinction may be made between algorithms that can be called **constructive** and those that can be described as **generate-and-test**. A constructive algorithm generates the content once, and is done with it; however, it needs to make sure that the content is correct or at least “good enough” as it is being constructed. An example of this approach is using fractals or cellular automata to generate terrains or grammars to generate levels (also refer to the corresponding PCG method sections below). A generate-and-test algorithm, instead, incorporates both a generate and a test mechanism. After a candidate content instance is generated, it is tested according to some criteria (e.g., is there a path between the entrance and exit of the dungeon, or does the tree have proportions within a certain range?). If the test fails, all or some of the candidate content is discarded and regenerated, and this process continues until the content is good enough. A popular PCG framework that builds upon the generate-and-test paradigm is the search-based [720] approach discussed in Section 4.3.

### 4.2.3 Taxonomy of Roles

In this section we identify and briefly outline the four possible roles a PCG algorithm can take in the game design process classified across the dimensions of autonomy and player-based adaptivity. The various PCG roles are illustrated in Fig. 4.2 and are extensively discussed in Section 4.4.

#### 4.2.3.1 Autonomy: Autonomous Versus Mixed-Initiative

The generative process that does not consider any input from the human designer is defined as **autonomous** PCG whereas **mixed-initiative** PCG refers to the process that involves the human designer in the creative task. Both roles are discussed in further detail in Section 4.4.

#### 4.2.3.2 Adaptivity: Experience-Agnostic Versus Experience-Driven

**Experience-agnostic** content generation refers to the paradigm of PCG where content is generated without taking player behavior or player experience into account, as opposed to **experience-driven** [783], adaptive, personalized or player-centered content generation where player interaction with the game is analyzed and content is created based on a player’s previous behavior. Most commercial games tackle PCG in a generic, experience-agnostic way, while experience-driven PCG has been receiving increasing attention in academia. Recent extensive reviews of PCG for player-adaptive games can be found in [783, 784].

### 4.3 How Could We Generate Content?

There are many different algorithmic approaches to generating content for games. While many of these methods are commonly thought of as AI methods, others are drawn from graphics, theoretical computer science or even biology. The various methods differ also in what types of content they are suitable to generate. In this section, we discuss a number of PCG methods that we consider important and include search-based, solver-based, and grammar-based methods but also cellular automata, noise and fractals.

#### 4.3.1 Search-Based Methods

The **search-based** approach to PCG [720] has been intensively investigated in academic PCG research in recent years. In search-based procedural content generation, an evolutionary algorithm or some other stochastic search or optimization algorithm is used to search for content with the desired qualities. The basic metaphor is that of design as a search process: a good enough solution to the design problem exists within some space of solutions, and if we keep iterating and tweaking one or many possible solutions, keeping those changes which make the solution(s) better and discarding those that are harmful, we will eventually arrive at the desired solution. This metaphor has been used to describe the design process in many different disciplines: for example, Will Wright—designer of *SimCity* (Electronic Arts, 1989) and *The Sims* (Electronic Arts, 2000)—described the game design process as search in his talk at the 2005 Game Developers Conference. Others have previously described the design process in general, and in other specialized domains such as architecture, the design process can be conceptualized as search and implemented as a computer program [757, 55].

The core components of the search-based approach to solving a content generation problem are the following:

- A **search algorithm**. This is the “engine” of a search-based method. Often relatively simple evolutionary algorithms work well enough, however sometimes there are substantial benefits to using more sophisticated algorithms that take e.g., constraints into account, or that are specialized for a particular content representation.
- A **content representation**. This is the representation of the artifacts you want to generate, e.g., levels, quests or winged kittens. The content representation could be anything from an array of real numbers to a graph to a string. The content representation defines (and thus also limits) what content can be generated, and determines whether effective search is possible; see also the discussion about representation in Chapter 2.
- One or more **evaluation functions**. An evaluation function is a function from an artifact (an individual piece of content) to a number indicating the quality of the artifact. The output of an evaluation function could indicate e.g., the playability of a level, the intricacy of a quest or the aesthetic appeal of a winged kitten. Crafting an evaluation function that reliably measures the aspect of game quality that it is meant to measure is often among the hardest tasks in developing a search-based PCG method. Refer also to the discussion about utility in Chapter 2.

Let us look at some of the choices for **content representations**. To take a very well-known example, a level in *Super Mario Bros* (Nintendo, 1985) might be represented in any of the following ways:

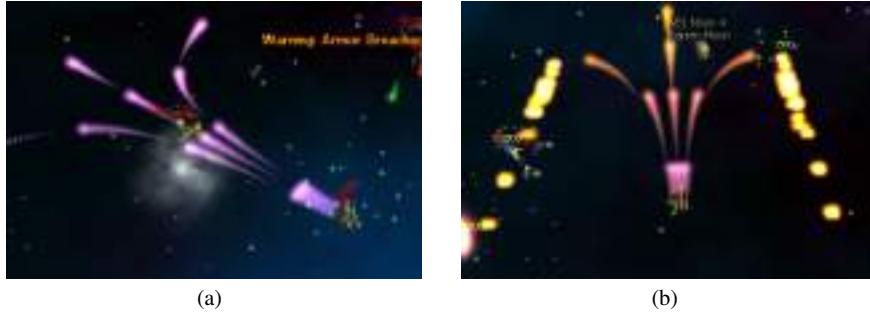
1. Directly, as a level map, where each variable in the genotype corresponds to one “block” in the phenotype (e.g., bricks, question mark blocks, etc.).
2. More indirectly, as a list of the positions and properties of the different game entities such as enemies, platforms, gaps and hills (an example of this can be found in [611]).
3. Even more indirectly, as a repository of different reusable patterns (such as collections of coins or hills), and a list of how they are distributed (with various transforms such as rotation and scaling) across the level map (an example of this can be found in [649]).
4. Very indirectly, as a list of desirable properties such as the number of gaps, enemies, or coins, the width of gaps, etc. (an example of this can be found in [617]).
5. Most indirectly, as a random number seed.

While it clearly makes no sense to evolve random number seeds (it is a representation with no locality whatsoever) the other levels of abstraction can all make sense under certain circumstances. The fundamental tradeoff is between more direct, more fine-grained representations with potentially higher locality (higher correlation of fitness between neighboring points in search space) and less direct, more coarse-grained representations with probably lower locality but smaller search spaces. Smaller search spaces are in general easier to search. However, larger search spaces would (all other things being equal) allow for more different types of content to be expressed, or in other words increase the *expressive range* of the generator.

The third important choice when designing a search-based PCG solution is the **evaluation function** known as the *fitness function*. The evaluation function assesses all candidate solutions, and assigns a score (a fitness value or evaluation value) to each candidate. This is essential for the search process; if we do not have a good evaluation function, the evolutionary process will not work as intended and will not find good content. It could be argued that designing an entirely accurate content quality evaluation is “AI-complete” in the sense that to really understand what a human finds fun, you actually have to be a human or understand humans in depth. However, like for other problems in various areas of AI, we can get pretty far with well-designed domain-specific heuristics. In general, the evaluation function should be designed to model some desirable quality of the artifact, e.g., its playability, regularity, entertainment value, etc. The design of an evaluation function depends to a great extent on the designer and what she thinks are the important aspects that should be optimized and how to formulate that.

In search-based PCG, we can distinguish between three classes of evaluation functions: direct, simulation-based, and interactive.

- **Direct** evaluation functions map the generated content (or features extracted from it) directly to a content quality value and, in that sense, they base their fitness calculations directly on the phenotype representation of the content. No simulation of the gameplay is performed during the mapping. Some direct evaluation functions are hand-coded, and some are learned from data. Direct evaluation functions are fast to compute and often relatively easy to implement, but it is sometimes hard to devise a direct evaluation function for some aspects of game content. Example features include the placement of bases and resources in real-time strategy games [712], the size of the ruleset in strategy games [415] or the current mood of the game scene based on visual attention theory [185]. The mapping between features and fitness might be contingent on a model of the playing style, preferences or affective state of players. An example of this form of fitness evaluation can be seen in the study done by Shaker et al. [617, 610] for personalizing player experience using models of players to give a measure of content quality. In that study, the authors trained neural networks to predict the player experience (such as challenge, frustration and enjoyment) of players given a playing style and the characteristics of a level as inputs. These trained neural networks can then be used as fitness functions by searching for levels that, for example, maximize predicted enjoyment while minimizing frustration. Or the exact opposite, if you wish.
- **Simulation-based** evaluation functions use AI agents that play through the generated content in order to estimate its quality. Statistics are calculated about the agents’ behavior and playing style and these statistics are then used to score game content. The type of the evaluation task determines the area of proficiency of the AI agent. If content is evaluated on the basis of playability, e.g., the existence of a path from the start to the end in a maze or a level in a 2D platform game, then AI agents should be designed that excel in reaching the end of the game. On the other hand, if content is optimized to maximize a particular player experience, then an AI agent that imitates human behavior should generally be adopted. An



**Fig. 4.3** Examples of evolved weapons in *Galactic Arms Races*. Images obtained from [www.aigameresearch.org](http://www.aigameresearch.org).

example study that implements a human-like agent for assessing content quality is presented in [704] where neural-network-based controllers are trained to drive like human players in a car racing game and then used to evaluate the generated tracks. Each track generated is given a fitness value according to statistics calculated while the AI controller is playing. Another example of a simulation-based evaluation function is measuring the average fighting time of bots in a first-person shooter game [103]. In that study, levels were simply selected to maximize the amount of time bots spent on the level before killing each other.

- **Interactive** evaluation functions evaluate content based on interaction with a human, so they require a human “in the loop”. Examples of this method can be found in the work by Hastings et al. [250], who implemented this approach by evaluating the quality of the personalized weapons evolved implicitly based on how often and how long the player chooses to use these weapons. Figure 4.3 presents two examples of evolved weapons for different players. Cardamone et al. [102] also used this form of evaluation to score racing tracks according to the users’ reported preferences. Ølsted et al. used the same approach to design levels for first-person shooter games [501]. The first case is an example of an *implicit* collection of data, i.e., that the players did not answer direct questions about their preferences, while players’ preferences were collected *explicitly* in the second. The problem with explicit data collection is that, if not well integrated, it requires the gameplay session to be interrupted. This method, however, provides a reliable and accurate estimator of player experience, as opposed to implicit data collection, which is usually noisy and based on assumptions. Hybrid approaches are sometimes employed to mitigate the drawbacks of these two methods by collecting information across multiple modalities such as combining player behavior with eye gaze and/or skin conductance. Example applications of this approach can be found in biofeedback-based camera viewpoint generation [434], level generation [610] and visuals generation in physical interactive games [771].

Search-based methods have extremely broad applicability, as evolutionary computation can be used to construct almost any kind of game content. However, this generality comes with a number of drawbacks. One is that it is generally a rather slow method, requiring the evaluation of a large number of candidate content items. The time required to evolve a good solution can also not be precisely predicted, as there is no runtime guarantee for evolutionary algorithms. This might make search-based PCG solutions unsuitable for time-critical content generation, such as when you only have a few seconds to serve up a new level in a game. It should also be noted that the successful application of search-based PCG methods relies on judicious design choices when it comes to the particular search algorithm, representation and evaluation function.

As we will see below, there are several algorithms that are generally better suited than evolutionary algorithms for generating game content of specific types. However, none of them have the **versatility** of the search-based approach, with abilities to incorporate all kinds of objectives and constraints. As we will see, many other algorithms for content generation can also be combined with search-based solutions, so that evolutionary algorithms can be used to search the parameter space of other algorithms.

### 4.3.2 Solver-Based Methods

While the search-based approach to content generation means using one or several objective functions, perhaps in conjunction with constraints, to specify for a randomized search algorithm such as an evolutionary algorithm what to look for, there is another approach to PCG which is also based on the idea of content generation as search in a space of artifacts. **Solver-based methods** for PCG use constraint solvers, such as those used in logic programming, to search for content artifacts that satisfy a number of constraints.

Constraint solvers allow you to specify a number of constraints in a specific language; some solvers require you to specify the constraints mathematically, others use a logic language for specification. Behind the scenes they can be implemented in many different ways. While there are some solvers that are based on evolutionary computation, it is more common to use specialized methods, such as reducing the problem to a SAT (satisfiability) problem and using a SAT solver to find solutions. Many such solvers progress not through evaluating whole solutions, but searching in spaces of partial solutions. This has the effect of iteratively pruning the search space: eliminating parts of the search space repeatedly until only viable solutions are left. This marks a sharp difference with the search-based paradigms, and also suggests some differences in the use cases for these classes of algorithms. For example, while evolutionary algorithms are *anytime algorithms*, i.e., they can be stopped at any point and some kind of solution will be available (though a better solution would probably have been available if you had the algorithm run for longer), this is not always the case with constraint satisfaction methods—though the time taken until a

viable solution is found can be very low, depending on how many constraints need to be satisfied. Unlike with evolutionary algorithms, it is possible to prove bounds on the worst-case complexity of SAT solvers and the algorithms that depend on them. However, while the worst-case complexity is often shockingly high, such algorithms can be very fast when applied (judiciously) in practice.

An example of solver-based methods is the level generator for Smith and Whitehead's *Tanagra* mixed-initiative platform level design tool [642]. At the core of the tool is a constraint-based generator of platform game levels. The constraint solver uses a number of constraints on what constitutes a solvable platform level (e.g., maximum jump length, and distance from jumps to enemies) as well as constraints based on aesthetic considerations, such as the "rhythm" of the level, to generate new platform game levels or level segments. This generation happens very fast and produces good results, but is limited to fairly linear levels. Another example is the work by El-Nasr et al. on procedural generation of lighting [188, 185]. The system developed in those studies configures and continuously modulates the lighting in a scene with the aim to enhance player experience by using constraint non-linear optimization to select the best lighting configuration.

One approach to constraint solving which can be particularly useful for PCG because of its generality is Answer Set Programming (ASP) [638]. ASP builds on AnsProlog, a constraint programming language which is similar to Prolog [69]. Complex sets of constraints can be specified in AnsProlog, and an ASP solver can then be used to find all models (all configurations of variables) that satisfy the constraints. For PCG purposes, the model (the set of parameters) can be used to describe a world, a story or similar, and the constraints can specify playability or various aesthetic considerations. An example of an ASP application for level and puzzle generation is the *Refraction* game (see Fig. 4.4). For a good introduction to and overview of the use of ASP in PCG you may refer to [638, 485].

Generally solver-based methods can be suitable when the whole problem can be encoded in the language of the constraint solver (such as AnsProlog). It is generally complicated to include simulation-based tests or any other call to the game engine inside a constraint satisfaction program. An alternative if simulation-based tests need to be performed is evolutionary algorithms, which can also be used to solve constraint satisfaction problems. This allows a combination of fitness values and constraints to drive evolution [376, 382, 240].

### 4.3.3 Grammar-Based Methods

Grammars are fundamental structures in computer science that also have many applications in procedural content generation. In particular, they are very frequently used for producing plants, such as trees, which are commonly used in many different types of games. However, grammars have also been used to generate missions and dungeons [173, 174], rocks [159], underwater environments [3] and caves [424]. In these cases, grammars are used as a constructive method, creating content with-



**Fig. 4.4** A screenshot from the *Refraction* educational game. A solver-based PCG method (ASP) is used to generate the levels and puzzles of the game. Further details about the application of ASP to *Refraction* can be found in [635, 89].

out any evaluation functions or re-generation. However, grammar methods can also be used together with search-based methods, so that the grammar expansion is used a genotype-to-phenotype mapping.

A (formal) **grammar** is a set of **production rules** for rewriting strings, i.e., turning one string into another. Each rule is of the form  $(\text{symbol(s)}) \rightarrow (\text{other symbol(s)})$ . Here are some example production rules:

1.  $A \rightarrow AB$
2.  $B \rightarrow b$

Expanding a grammar is as simple as going through a string, and each time a symbol or sequence of symbols that occurs in the *left-hand side* of a rule is found, those symbols are replaced by the *right-hand side* of that rule. For example, if the initial string is  $A$ , in the first rewriting step the  $A$  would be replaced by  $AB$  by rule 1, and the resulting string will be  $AB$ . In the second rewriting step, the  $A$  would again be transformed to  $AB$  and the  $B$  would be transformed to  $b$  using rule 2, resulting in the string  $ABb$ . The third step yields the string  $ABbb$  and so on. A convention in grammars is that upper-case characters are nonterminal symbols, which are on the left-hand side of rules and therefore rewritten further, whereas lower-case characters are terminal symbols which are not rewritten further.

Starting with the axiom A (in L-systems the seed strings are called axioms) the first few expansions look as follows:

```
A
AB
ABA
ABAAB
ABAABABA
ABAABABAABAAB
ABAABABAABAABABAABABA
ABAABABAABAABABAABAABABAAB
```

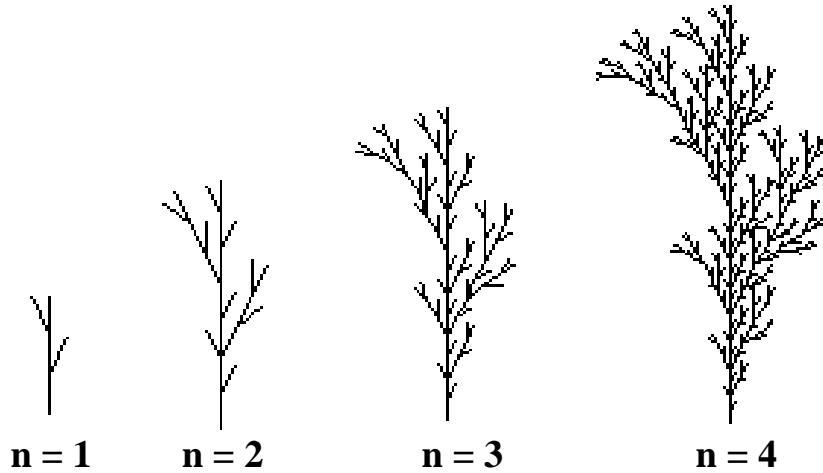
This particular grammar is an example of an **L-system**. L-systems are a class of grammars whose defining feature is parallel rewriting, and was introduced by the biologist Aristid Lindenmayer in 1968 explicitly to model the growth of organic systems such as plants and algae [387]. With time, they have turned out to be very useful for generating plants in games as well as in theoretical biology.

One way of using the power of L-systems to generate 2D (and 3D) artifacts is to interpret the generated strings as instructions for a turtle in **turtle graphics**. Think of the turtle as moving across a plane holding a pencil, and simply drawing a line that traces its path. We can give commands to the turtle to move forwards, or to turn left or right. For example, we can define the L-system alphabet  $\{F, +, -, [, ]\}$  and then use the following key to interpret the generated strings:

- F: move forward a certain distance (e.g., 10 pixels).
- +: turn left 30 degrees.
- -: turn right 30 degrees.
- [: push the current position and orientation onto the stack.
- ]: pop the position and orientation off the stack.

Bracketed L-systems can be used to generate surprisingly plant-like structures. Consider the L-system defined by the single rule  $F \rightarrow F[-F]F[+F][F]$ . Figure 4.5 shows the graphical interpretation of the L-system after 1, 2, 3 and 4 rewrites starting from the single symbol  $F$ . Minor variations of the rule in this system generate different but still plant-like structures, and the general principle can easily be extended to three dimensions by introducing symbols that represent rotation along the axis of drawing. For this reason, many standard packages for generating plants in game worlds are based on L-systems or similar grammars. For a multitude of beautiful examples of plants generated by L-systems refer to the book *The Algorithmic Beauty of Plants* by Prusinkiewicz and Lindenmayer [542].

There are many extensions of the basic L-system formalism, including non-deterministic L-systems, which can help with increasing diversity of the generated content, and context-sensitive L-systems, which can produce more complicated patterns. Formally specifying L-systems can be a daunting task, in particular as the mapping between the axiom and rules on the one hand and the results after expansion on the other are so complex. However, search-based methods can be used to



**Fig. 4.5** Four rewrites of the bracketed L-system  $F \rightarrow F[-F]F[+F][F]$ .

find good axioms or rules, using for example desired height or complexity of the plant in the evaluation function [498].

#### 4.3.4 Cellular Automata

A cellular automaton (plural: **cellular automata**) is a discrete model of computation which is widely studied in computer science, physics, complexity science and even some branches of biology, and can be used to computationally model biological and physical phenomena such as growth, development, patterns, forms, or even emergence. While cellular automata (CA) have been the subject of extensive study, the basic concept is actually very simple and can be explained in a sentence or two: cellular automata are a set of cells placed on a grid that change through a number of discrete time steps according to a set of rules; these rules rely on the current state of each cell and the state of its neighboring cells. The rules can be applied iteratively for as many time steps as we desire. The conceptual idea behind cellular automata was introduced by Stanislaw Ulam and John von Neumann [742, 487] back in the 1940s; it took about 30 more years, however, for us to see an application of cellular automata that showed their potential beyond basic research. That application was the two-dimensional cellular automaton designed in Conway's *Game of Life* [134]. The *Game of Life* is a zero-player game; its outcome is not influenced by the player's input throughout the game and it is solely dependent on its initial state (which is determined by the player).

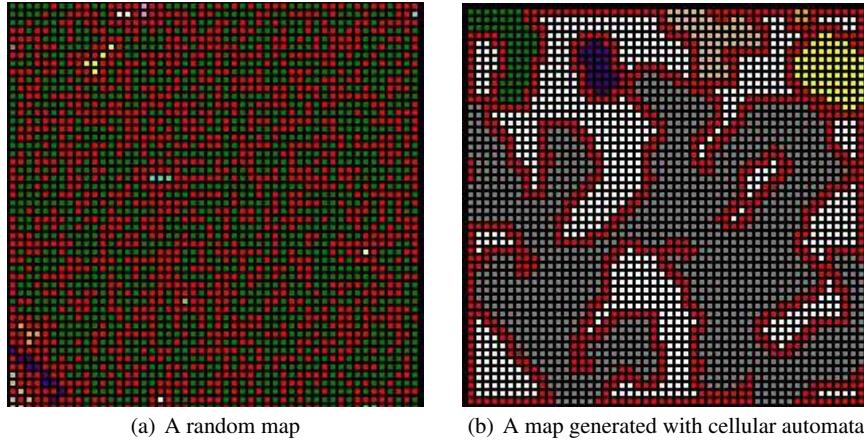
A cellular automaton contains a number of cells represented in any number of dimensions; most cellular automata, however, are either one-dimensional (vectors) or two-dimensional (matrices). Each cell can have a finite number of **states**; for instance, the cell can be *on* or *off*. A set of cells surrounding each cell define its **neighborhood**. The neighborhood defines which cells around a particular cell affect the cell's future state and its size can be represented by any integer number greater than 1. For one-dimensional cellular automata, for instance, the neighborhood is defined by the number of cells to the left or the right of the cell. For two-dimensional cellular automata, the two most common neighborhood types are the **Moore** and the **von Neumann** neighborhood. The former neighborhood type is a square consisting of the cells surrounding a cell, including those surrounding it diagonally; for example, a Moore neighborhood of size 1 contains the eight cells surrounding each cell. The latter neighborhood type, instead, forms a cross of cells which are centered on the cell considered. For example, a von Neumann neighborhood of size 1 consists of the four cells surrounding the cell, above, below, to the left and to the right.

At the beginning of an experiment (at time  $t = 0$ ) we initialize the cells by assigning a state for each one of them. At each time step  $t$  we create a new generation of cells according to a rule or a mathematical function which specifies the new state of each cell given the current state of the cell and the states of the cells in its neighborhood at time  $t - 1$ . Normally, the rule for updating the state of the cells remains the same across all cells and time steps (i.e., it is static) and is applied to the whole grid.

Cellular automata have been used extensively in games for modeling environmental systems like heat and fire, rain and fluid flow, pressure and explosions [209, 676] and in combination with influence maps for agent decision making [678, 677]. Another use for cellular automata has been for thermal and hydraulic erosion in procedural terrain generation [500]. Of particular interest for the purposes of this section is the work of Johnson et al. [304] on the generation of infinite cave-like dungeons using cellular automata. The motivation in that study was to create an infinite cave-crawling game, with environments stretching out endlessly and seamlessly in every direction. An additional design constraint was that the caves are supposed to look organic or eroded, rather than having straight edges and angles. No storage medium is large enough to store a truly endless cave, so the content must be generated at runtime, as players choose to explore new areas. The game does not scroll but instead presents the environment one screen at a time, which offers a time window of a few hundred milliseconds in which to create a new room every time the player exits a room.

The method introduced by Johnson et al. [304] used the following four parameters to control the map generation process:

- a percentage of rock cells (inaccessible areas) at the beginning of the process;
- the number of CA generations (iterations);
- a neighborhood threshold value that defines a rock;
- the Moore neighborhood size.

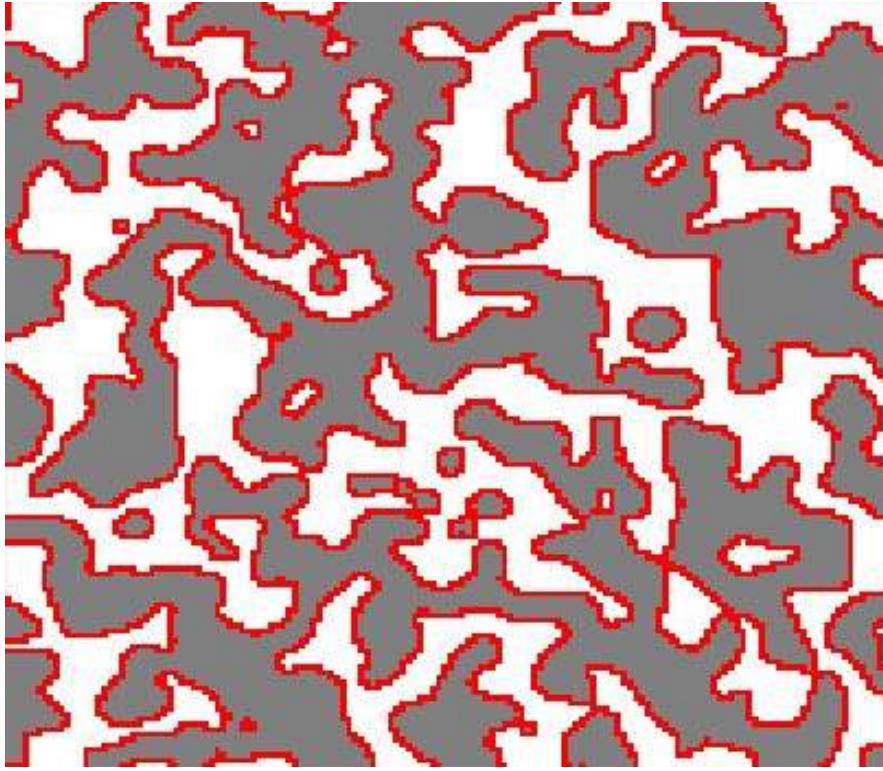


**Fig. 4.6** Cave generation: Comparison between a CA and a randomly generated map. The CA parameters used are as follows: the CA runs for four generations; the size of the Moore neighborhood considered is 1; the threshold value for the CA rule is 5 ( $T = 5$ ); and the percentage of rock cells at the beginning of the process is 50% (for both maps). Rock and wall cells are represented by white and red color respectively. Colored areas represent different tunnels (floor clusters). Images adapted from [304].

In the dungeon generation implementation presented in [304], each room is a  $50 \times 50$  grid, where each cell can be in one of two states: *empty* or *rock*. Initially, the grid is empty. The generation of a single room is as follows.

1. The grid is “sprinkled” with rocks: for each cell, there is probability (e.g., 0.5) that it is turned into rock. This results in a relatively uniform distribution of rock cells.
2. A number of CA generations (iterations) are applied to this initial grid.
3. For each generation the following simple rule is applied to all cells: a cell turns into rock in the next iteration if at least  $T$  (e.g., 5) of its neighbors are rock, otherwise it will turn into free space.
4. For aesthetic reasons the rock cells that border on empty space are designated as “wall” cells, which are functionally rock cells but look different.

The aforementioned simple procedure generates a surprisingly lifelike cave-room. Figure 4.6 shows a comparison between a random map (sprinkled with rocks) and the results of a few iterations of the cellular automaton. But while this process generates a single room, a game would normally require a number of connected rooms. A generated room might not have any openings in the confining rocks, and there is no guarantee that any exits align with entrances to the adjacent rooms. Therefore, whenever a room is generated, its immediate neighbors are also generated. If there is no connection between the largest empty spaces in the two rooms, a tunnel is drilled between those areas at the point where they are least separated. A few more iterations of the CA algorithm are then run on all nine neighboring rooms



**Fig. 4.7** Cave generation: a  $3 \times 3$  base grid map generated with CA. Rock and wall cells are represented by white and red color respectively; gray areas represent floor. Moore neighborhood size is 2,  $T$  is 13, number of CA iterations is 4, and the percentage of rock cells at the initialization phase is 50%. Image adapted from [304].

together, to smooth out any sharp edges. Figure 4.7 shows the result of this process, in the form of nine rooms that seamlessly connect. This generation process is extremely fast, and can generate all nine rooms in less than a millisecond on a modern computer. A similar approach to that of Johnson et al. is featured in the *Galak-Z* (17-bit, 2016) game for dungeon generation [9]. In that game cellular automata generate the individual rooms of levels and the rooms are tied together via a variation of a Hilbert curve, which is a continuous fractal space-filling curve [261]. *Galak-Z* (17-bit, 2016) shows an alternative way of combining CA with other methods for achieving the desired map generation result.

In summary, CA are very fast constructive methods that can be used effectively to generate certain kinds of content such as terrains and levels (as e.g., in [304]), but they can also be potentially used to generate other types of content. The greatest benefit a CA algorithm can offer to a game content generator is that it depends on a small number of parameters and that it is intuitive and relatively simple to grasp and implement. However, the algorithm's constructive nature is the main cause for

its disadvantages. For both designers and programmers, it is not trivial to fully understand the impact that a single parameter may have on the generation process, since each parameter affects multiple features of the generated output. While the few parameters of the core algorithm allow for a certain degree of controllability, the algorithm cannot guarantee properties such as playability or solvability of levels. Further, it is not possible to design content that has specific requirements, e.g., a map with a certain connectivity, since gameplay features are disjoint from the control parameters of the CA. Thus, any link between the CA generation method and gameplay features would have to be created through a process of trial and error. In other words, one would need to resort to preprocessing or a generate-and-test approach.

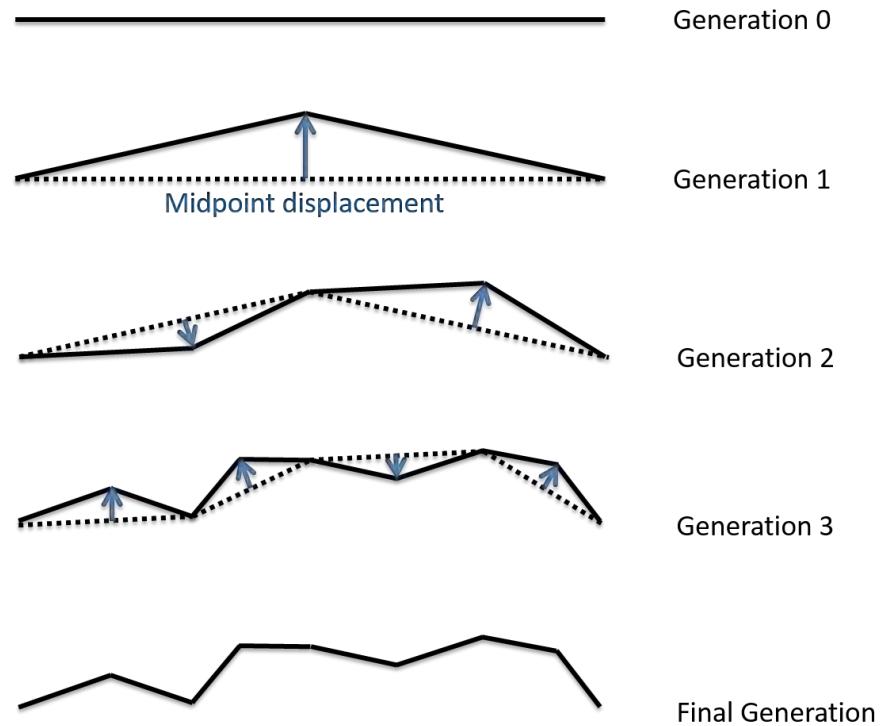
### 4.3.5 Noise and Fractals

One class of algorithms that are very frequently used to generate heightmaps and textures are **noise** algorithms, many of which are **fractal** algorithms, meaning that they exhibit scale-invariant properties. Noise algorithms are usually fast and easy to use but lack in controllability.

Both textures and many aspects of terrains can fruitfully be represented as two-dimensional matrices of real numbers. The width and height of the matrix map to the  $x$  and  $y$  dimensions of a rectangular surface. In the case of a texture, this is called an **intensity map**, and the values of cells correspond directly to the brightness of the associated pixels. In the case of terrains, the value of each cell corresponds to the height of the terrain (over some baseline) at that point. This is called a **heightmap**. If the resolution with which the terrain is rendered is greater than the resolution of the heightmap, intermediate points on the ground can simply be interpolated between points that do have specified height values. Thus, using this common representation, any technique used to generate noise could also be used to generate terrains, and vice versa—though they might not be equally suitable.

It should be noted that in the case of terrains, other representations are possible and occasionally suitable or even necessary. For example, one could represent the terrain in three dimensions, by dividing the space up into *voxels* (cubes) and computing the three-dimensional voxel grid. An example is the popular open-world game *Minecraft* (Mojang, 2011), which uses unusually large voxels. Voxel grids allow structures that cannot be represented with heightmaps, such as caves and overhanging cliffs, but they require a much larger amount of storage.

Fractals [180, 500] such as **midpoint displacement algorithms** [39] are in common use for real-time map generation. Midpoint displacement is a simple algorithm for generating two-dimensional landscapes (seen from the side) by repeatedly subdividing a line. The procedure is as follows: start with a horizontal line. Find the midpoint of that line, and move the line up or down by a random amount, thus breaking the line in two. Now do the same thing for both of the resulting lines, and so on for as many steps as you need in order to reach sufficient resolution. Ev-



**Fig. 4.8** The Midpoint Displacement algorithm visualized.

every time you call the algorithm recursively, lower the range of the random number generator somewhat (see Fig. 4.8 for an example).

A useful and simple way of extending the midpoint displacement idea to two dimensions (and thus creating two-dimensional heightmaps which can be interpreted as three-dimensional landscapes) is the **Diamond-Square algorithm** (also known as “the cloud fractal” or “the plasma fractal” because of its frequent use for creating such effects) [210]. This algorithm uses a square 2D matrix with width and height  $2^n + 1$ . To run the algorithm you normally initialize the matrix by setting the values of all cells to 0, except the four corner values which are set to random values in some chosen range (e.g.,  $[-1, 1]$ ). Then you perform the following steps:

1. *Diamond step:* Find the midpoint of the four corners, i.e., the most central cell in the matrix. Set the value of that cell to the average value of the corners. Add a random value to the middle cell.

2. *Square step:* Find the four cells in between the corners. Set each of those to the average value of the two corners surrounding it. Add a random value to each of these cells.

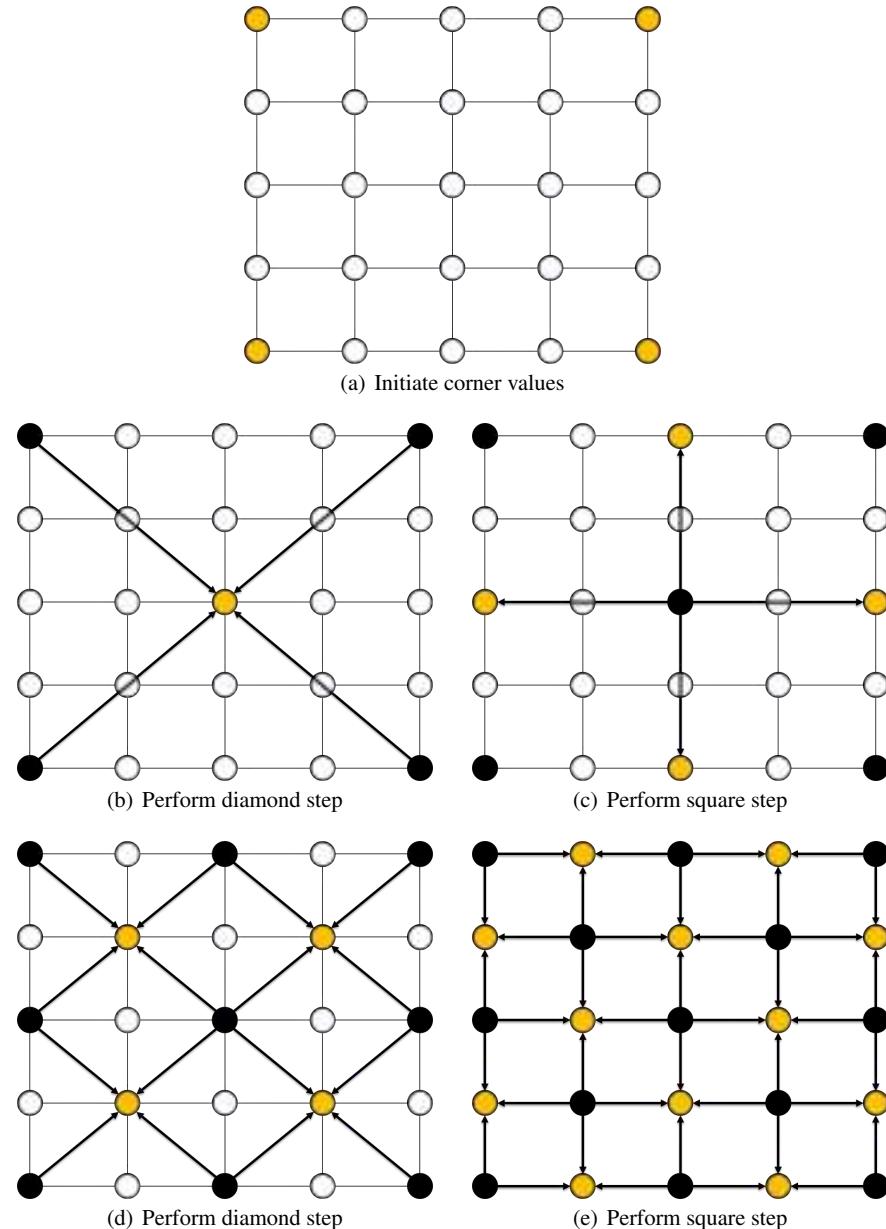
Call this method recursively for each of the four subsquares of the matrix, until you reach the resolution limit of the matrix ( $3 \times 3$  sub-squares). Every time you call the method, reduce the range of the random values somewhat. The process is illustrated in Fig. 4.9.

There are many more advanced methods for generating fractal noise, with different properties. One of the most important is **Perlin noise**, which has some benefits over Diamond Square [529]. These algorithms are covered thoroughly in books that focus on texturing and modeling from a graphics perspective [180].

#### 4.3.6 Machine Learning

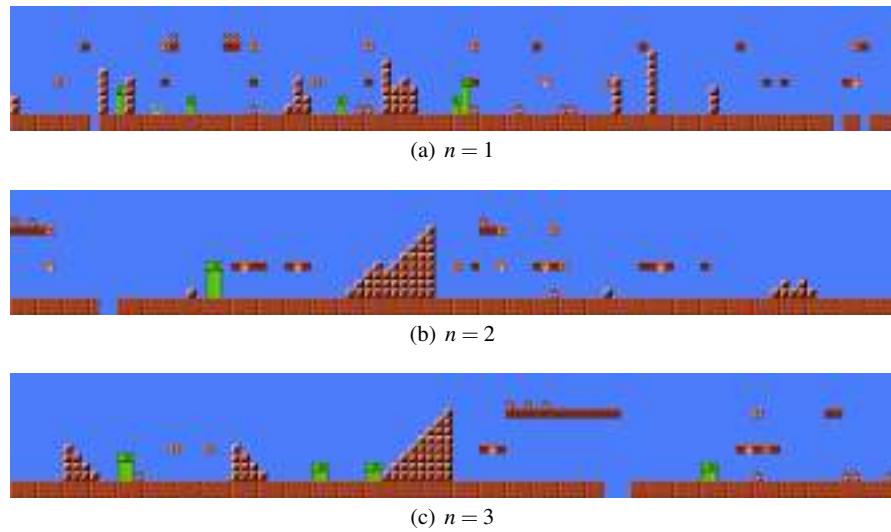
An emerging direction in PCG research is to train generators on **existing content**, to be able to produce more content of the same type and style. This is inspired by the recent results in deep neural networks, where network architectures such as **generative adversarial networks** [232] and **variational autoencoders** [342] have attained good results in learning to produce images of e.g., bedrooms, cats or faces, and also by earlier results where both simpler learning mechanisms such as Markov chains and more complex architectures such as recurrent neural networks have learned to produce text and music after training on some corpus.

While these kinds of generative methods based on machine learning work well for some types of content—most notably music and images—many types of game content pose additional challenges. In particular, a key difference between game content generation and procedural generation in many other domains is that most game content has strict structural constraints to ensure playability. These constraints differ from the structural constraints of text or music because of the need to play games in order to experience them. A level that structurally prevents players from finishing it is not a good level, even if it is visually attractive; a strategy game map with a strategy-breaking shortcut will not be played even if it has interesting features; a game-breaking card in a collectible card game is merely a curiosity; and so on. Thus, the domain of game content generation poses different challenges from that of other generative domains. The same methods that can produce “mostly correct” images of bedrooms and horses, that might still have a few impossible angles or vestigial legs, are less suitable for generating mazes which must have an exit. This is one of the reasons why machine learning-based approaches have so far only attained limited success in PCG for games. The other main reason is that for many types of game content, there simply isn’t enough existing content to train on. This



**Fig. 4.9** The Diamond-Square algorithm visualized in five steps. Adapted from a figure by Christopher Ewin, licensed under CC BY-SA 4.0.

is, however, an active research direction where much progress might be achieved in the next few years.



**Fig. 4.10** Mario levels reconstructed by  $n$ -grams with  $n$  set to 1, 2, and 3, respectively.

The core difference between PCG via machine learning and approaches such as search-based PCG is that the content is created *directly* (e.g., via sampling) from models which have been trained on game content. While some search-based PCG approaches use evaluation functions that have been trained on game content—for instance, the work of Shaker et al. [621] or Liapis et al. [373]—the actual content generation is still based on search. Below, we present some examples of PCG via machine learning; these particular PCG studies built on the use of  **$n$ -grams**, **Markov models** and **artificial neural networks**. For more examples of early work along these lines, see the recent survey paper [668].

#### 4.3.6.1 $n$ -grams and Markov Models

For content that can be expressed as one- or two-dimensional discrete structures, such as many game levels, methods based on Markov models can be used. One particularly straightforward Markov model is the  $n$ -gram model, which is commonly used for text prediction. The  $n$ -gram method is very simple—essentially, you build conditional probability tables from strings and sample from these tables when constructing new strings—and also very fast.

Dahlskog et al. trained  $n$ -gram models on the levels of the original *Super Mario Bros* (Nintendo, 1985) game, and used these models to generate new levels [156]. As  $n$ -gram models are fundamentally one-dimensional, these levels needed to be converted to strings in order for  $n$ -grams to be applicable. This was done through dividing the levels into vertical “slices,” where most slices recur many times throughout the level [155]. This representational trick is dependent on there being a large

amount of redundancy in the level design, something that is true in many games. Models were trained using various levels of  $n$ , and it was observed that while  $n = 0$  creates essentially random structures and  $n = 1$  creates barely playable levels,  $n = 2$  and  $n = 3$  create rather well-shaped levels. See Fig. 4.10 for examples of this.

Summerville et al. [667] extended these models with the use of Monte Carlo tree search to guide the generation. Instead of solely relying on the learned conditional probabilities, they used the learned probabilities during rollouts (generation of whole levels) that were then scored based on an objective function specified by a designer (e.g., allowing them to bias the generation towards more or less difficult levels). The generated levels could still only come from observed configurations, but the utilization of MCTS meant that playability guarantees could be made and allowed for more designer control than just editing of the input corpus. This can be seen as a hybrid between a search-based method and a machine learning-based method. In parallel, Snodgrass and Ontañón trained two-dimensional Markov Chains—a more complex relative of the  $n$ -gram—to generate levels for both *Super Mario Bros* (Nintendo, 1985) and other similar platform games, such as *Lode Runner* (Brøderbund, 1983) [644].

#### 4.3.6.2 Neural Networks

Given the many uses of **neural networks** in machine learning, and the many different neural network architectures, it is little wonder that neural networks are also highly useful for machine learning-based PCG. Following on from the *Super Mario Bros* (Nintendo, 1985) examples in the previous section, Hoover et al. [277] generated levels for that same game by extending a representation called functional scaffolding for musical composition (FSMC) that was originally developed to compose music. The original FSMC representation posits 1) music can be represented as a function of time and 2) musical voices in a given piece are functionally related [276]. Through a method for evolving neural networks called NeuroEvolution of Augmenting Topologies [655], additional musical voices are evolved to be played simultaneously with an original human-composed voice. To extend this musical metaphor and represent *Super Mario Bros* (Nintendo, 1985) levels as functions of time, each level is broken down into a sequence of tile-width columns. The height of each column extends the height of the screen. While FSMC represents a unit of time by the length of an eighth-note, a unit of time in this approach is the width of each column. At each unit of time, the system queries the ANN to decide a height to place a tile. FSMC then inputs a musical note's pitch and duration to the ANNs. This approach translates pitch to the height at which a tile is placed and duration to the number of times a tile-type is repeated at that height. For a given tile-type or musical voice, this information is then fed to a neural network that is trained on two-thirds of the existing human-authored levels to predict the value of a tile-type at each column. The idea is that the neural network will learn hidden relationships between the tile-types in the human-authored levels that can then help humans construct entire levels from as little starting information as the layout of a single tile-type.

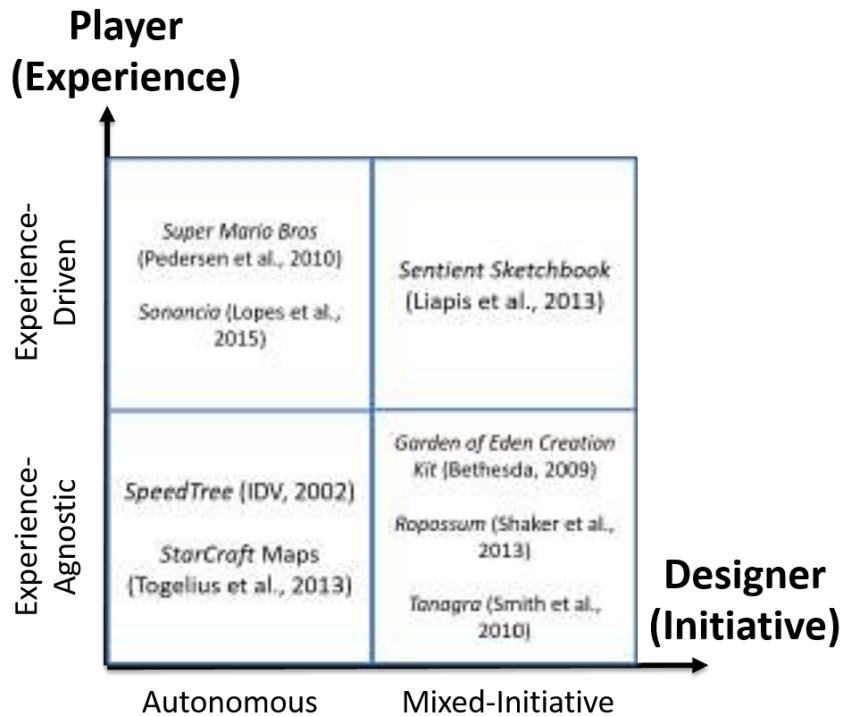
Of course, machine learning can also be used to generate other types of game content that are not levels. A fascinating example of this is *Mystical Tutor*, a design assistant for *Magic: The Gathering* cards [666]. In contrast to some of the other generators that aim to produce complete, playable levels, *Mystical Tutor* acknowledges that its output is likely to be flawed in some ways and instead aims to provide inspirational raw material for card designers.

## 4.4 Roles of PCG in Games

The generation of content algorithmically may take different roles within the domain of games. We can identify two axes across which PCG roles can be placed: players and designers. We envision PCG systems that consider designers while they generate content or they operate interdependently of designers; the same applies for players. Figure 4.11 visualizes the key roles of PCG in games across the dimensions of designer initiative and player experience.

Regardless of the generation method used, game genre or content type PCG can act either *autonomously* or as a *collaborator* in the design process. We refer to the former role as **autonomous** generation (Section 4.4.2) and the latter role as **mixed-initiative** (Section 4.4.1) generation. Further, we cover the **experience-driven** PCG role by which PCG algorithms consider the player experience in whatever they try to generate (Section 4.4.3). As a result, the generated content is associated to the player and her experience. Finally, if PCG does not consider the player as part of the generation process it becomes **experience-agnostic** (Section 4.4.4).

PCG techniques can be used to generate content in **runtime**, as the player is playing the game, allowing the generation of endless variations, making the game infinitely replayable and opening the possibility of generating player-adapted content, or **offline** during the development of the game or before the start of a game session. The use of PCG for offline content generation is particularly useful when generating complex content such as environments and maps; several examples of that was discussed at the beginning of the chapter. An example of the use of runtime content generation can be found in the game *Left 4 Dead* (Valve, 2008), a first-person shooter game that provides dynamic experience for each player by analyzing player behavior on the fly and altering the game state accordingly using PCG techniques [14, 60]. A trend related to runtime content generation is the creation and sharing of player-generated content. Some games such as *LittleBigPlanet* (Sony Computer Entertainment, 2008) and *Spore* (Electronic Arts, 2008) provide a content editor (level editor in the case of *LittleBigPlanet* and the *Spore* Creature Creator) that allows the players to edit and upload complete creatures or levels to a central online server where they can be downloaded and used by other players. With respect to the four different roles of PCG in games, runtime generation is possible in the autonomous and experience-agnostic roles, it is always the case in the experience-driven role whereas it is impossible in the mixed-initiative role. On the other hand, the offline generation of content can occur both autonomously and in



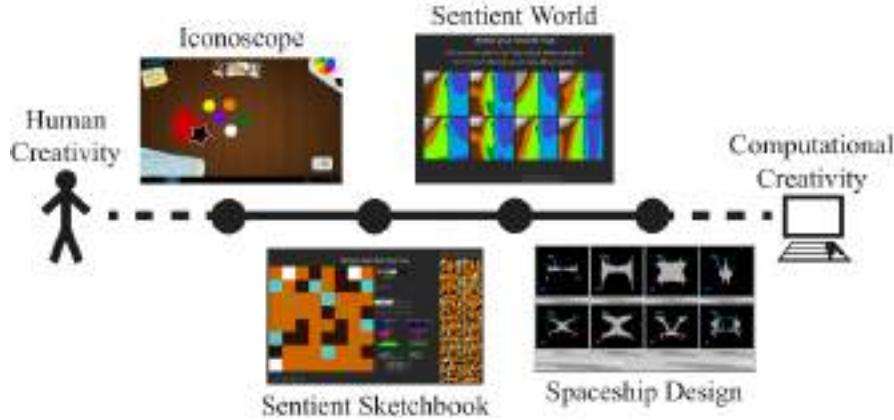
**Fig. 4.11** The four key PCG roles in games across the dimensions of designer initiative and player experience. For each combination of roles the figure lists a number of indicative examples of tools or studies covered in this chapter.

an experience-agnostic manner. Further it is exclusively the only way to generate content in a mixed-initiative fashion whereas it is not relevant for experience-driven generation as this PCG role occurs in runtime by definition.

The following four subsections outline the characteristics of each of the four PCG roles with a particular emphasis on the mixed-initiative and the experience-driven roles that have not yet covered in length in this chapter.

#### 4.4.1 Mixed-Initiative

**AI-assisted game design** refers to the use of AI-powered tools to support the game design and development process. This is perhaps the AI research area which is most promising for the development of better games [764]. In particular, AI can assist in the creation of game content varying from levels and maps to game mechanics and narratives.



**Fig. 4.12** The mixed-initiative spectrum between human and computer initiative (or creativity) across a number of mixed-initiative design tools discussed in this section. *Iconoscope* is a mixed-initiative drawing game [372], *Sentient Sketchbook* is a level editor for strategy games [379], *Sentient World* is mixed-initiative map editor [380] and *Spaceship Design* is a mixed-initiative (mostly computer initiative) tool powered by interactive evolution [377]. Adapted from [375].

We identify AI-assisted game design as the task of creating artifacts via the interaction of a human initiative and a computational initiative. The computational initiative is a PCG process and thus, we discuss this co-design approach under the heading of procedural content generation. Although the term **mixed-initiative** lacks a concrete definition [497], in this book we define it as the process that considers both the human and the computer *proactively* making content contributions to the game design task although the two initiatives do not need to contribute to the same degree [774]. Mixed-initiative PCG thus differs from other forms of co-creation, such as the collaboration of multiple human creators or the interaction between a human and non-proactive computer support tools (e.g., spell-checkers or image editors) or non-computer support tools (e.g., artboards or idea cards). The initiative of the computer can be seen as a continuum between the *no initiative* state, leaving full control of the design to the designer and having the computer program simply carry out the commands of the human designer, to the *full initiative* state which yields an autonomously creative system. Any state between the two is also possible as we will see in the examples below and as depicted in Fig. 4.12.

#### 4.4.1.1 Game Domains

While the process of AI-assisted game design is applicable to any creative facets within game design [381] it is level design that has benefited the most from it. Within commercial-standard game development, we can find AI-based tools that allow varying degrees of computer initiative. On one end of the scale, level editors such as the *Garden of Eden Creation Kit* (Bethesda, 2009) or game engines such

as the *Unreal Development Kit* (Epic Games 2009) leave most of creative process to the designer but they, nevertheless, boost game development through automating interpolations, pathfinding and rendering [774]. On the other end of the computer initiative scale, PCG tools specialized on e.g., vegetation—*SpeedTree* (IDV, 2002)—or FPS levels—*Oblige* (Apted, 2007)—only require the designer to set a small amount of generation parameters and, thus, the generation process is almost entirely autonomous.

Within academia, the area of AI-assisted game design tools has seen significant research interest in recent years [785] with contributions mainly to the level design task across several game genres including platformers [641], strategy games [380, 379, 774, 378] (see Fig. 4.13(a)), open world games [634], racing games [102], casual puzzle games [614] (see Fig. 4.13(b)), horror games [394], first-person shooters [501], educational games [89, 372], mobile games [482], and adventure games [323]. The range of mixed-initiative game design tools expands to tools that are designed to assist with the generation of complete game rulesets such as the MetaGame [522], the RuLearn [699] and the Ludocore [639] tools to tools that are purposed to generate narratives [480, 673] and stories within games [358].

#### 4.4.1.2 Methods

Any PCG approach could potentially be utilized for mixed-initiative game design. The dominant methods that have so far been used, however, rely on evolutionary computation following the search-based PCG paradigm. Even though evolution, at first sight, does not appear to be the most attractive approach for real-time processing and generation of content, it offers great advantages associated, in particular, with the stochasticity of artificial evolution, diversity maintenance and potential for balancing multiple design objectives. Evolution can be constrained to the generation of playable, usable, or, in general, content of particular *qualities* within desired design specifications. At the same time, it can incorporate metrics such as novelty [382] or surprise [240], for maximizing the *diversity* of generated content and thus enabling a change in the creative path of the designer [774]. Evolution can be computationally costly, however, and thus, interactive evolution is a viable and popular alternative for mixed-initiative evolutionary-based generation (e.g., see [102, 380, 377, 501]).

Beyond artificial evolution, another class of algorithms that is relevant for mixed-initiative content generation is constraint solvers and constraint optimization. Methods such as answer set programming [383, 69] have been used in several AI-assisted level design tools including *Tanagra* [641] for platformers and *Refraction* [89] for educational puzzle games. Artificial neural networks can also perform certain tasks in a mixed-initiative manner, such as performing “autocomplete” or level repair [296] through the use of deep learning approaches such as stacked autoencoders. The goal here is to provide a tool that “fills in” parts of a level that the human designer does not want or have time to create, and correcting other parts to achieve further consistency.

(a) *Sentient Sketchbook*(b) *Ropossum*

**Fig. 4.13** Examples of mixed-initiative level design tools. *Sentient Sketchbook* (a) offers map sketch suggestions to the designer via artificial evolution (see rightmost part of the image); the suggestions are evolved to either maximize particular objectives of the map (e.g., balance) or are evolved to maximize the novelty score of the map. In *Ropossum* (b) the designer may select to design elements of *Cut the Rope* (Chillingo, 2010) game levels; the generation of the remaining elements are left to evolutionary algorithms to design.

#### 4.4.2 Autonomous

The role of autonomous generation is arguably the most dominant PCG role in games. The earlier parts of this chapter are already dedicated to extensive discussions and studies of PCG systems that do not consider the designer in their creative process. As a result we will not cover this PCG role in further detail here. What is important to be discussed, however, is the fuzzy borderline between mixed-initiative and autonomous PCG systems. It might be helpful, for instance, to consider autonomous PCG as the process by which the role of the designer starts and ends with an offline setup of the algorithm. For instance, the designer is only involved in the parameterization of the algorithm as in the case of *SpeedTree* (IDV, 2002). One might wish, however, to further push the borderline between autonomous and mixed-initiative generation and claim that generation is genuinely autonomous only if the creative process reconsiders and adapts the utility function that drives the content generation—thereby becoming creative in its own right. A static utility function that drives the generation is often referred to as *mere* generation within the computational creativity field [381].

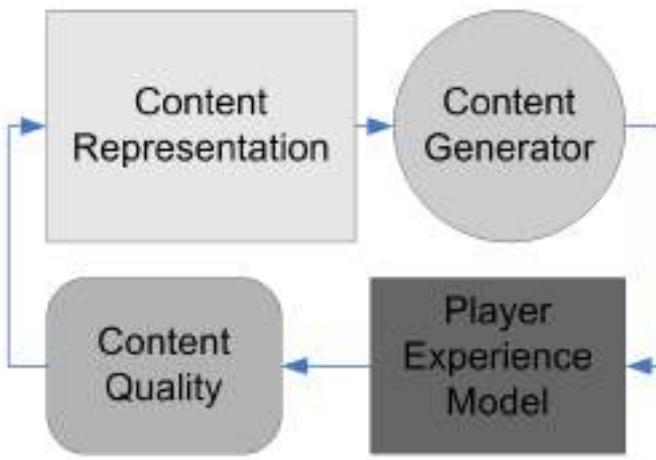
While the line between autonomy and collaboration with designers is still an open research question, for the purposes of this book, we can safely claim that the PCG process is autonomous when the initiative of the designer is limited to algorithmic parameterizations before the generation starts.

#### 4.4.3 Experience-Driven

As games offer one of the most representative examples of rich and diverse content creation applications and are elicitors of unique user experiences **experience-driven** PCG (EDPCG) [783, 784] views game content as the *building block* of games and the generated games as the potentiators of player experience. Based on the above, EDPCG is defined as a generic and effective approach for the optimization of user (player) experience via the adaptation of the experienced content. According to the experience-driven role of PCG in games *player experience* is the collection of affective patterns elicited, cognitive processes emerged and behavioral traits observed during gameplay [781].

By coupling player experience with procedural content generation, the experience-driven perspective offers a new, player-centered role to PCG. Since games are composed by game content that, when played by a particular player, elicits experience patterns, one needs to assess the quality of the content generated (linked to the experience of the player), search through the available content, and generate content that optimizes the experience for the player (see Fig. 4.14). In particular, the key components of EDPCG are:

- **Player experience modeling:** player experience is modeled as a function of game content and player.



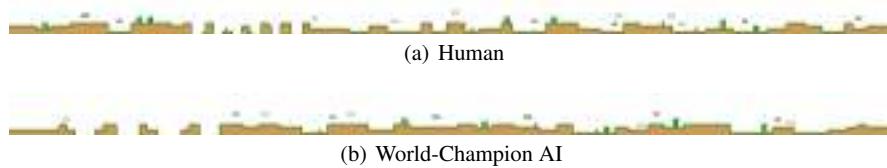
**Fig. 4.14** The four key components of the experience-driven PCG framework.

- **Content quality:** the quality of the generated content is assessed and linked to the modeled experience.
- **Content representation:** content is represented accordingly to maximize search efficacy and robustness.
- **Content generator:** the generator searches through the generative space for content that optimizes the experience for the player according to the acquired model.

Each component of EDPCG has its own dedicated literature and the extensive review of each is covered in other parts of the book. In particular, player experience modeling is covered in Chapter 5 whereas the remaining three components of the framework have already been covered in this chapter. A detailed survey and discussion about EDPCG is available in [783].

#### 4.4.3.1 Experience-Driven PCG in Practice

*Left 4 Dead* (Valve, 2008) is an example of the use of experience-driven PCG in a commercial game where an algorithm is used to adjust the pacing of the game on the fly based on the player's *emotional intensity*. In this case, adaptive PCG is used to adjust the difficulty of the game in order to keep the player engaged [60]. Adaptive content generation can also be used with another motive such as the generation of more content of the kind the player seems to like. This approach was followed, for instance, in the *Galactic Arms Race* [250] game where the weapons presented to the player are evolved based on her previous weapon use and preferences. In another EDPCG study, El-Nasr et al. implemented a direct fitness function—derived from visual attention theory—for the procedural generation of lighting [188, 185]. The procedural *Zelda* game engine [257], a game engine designed to emulate the

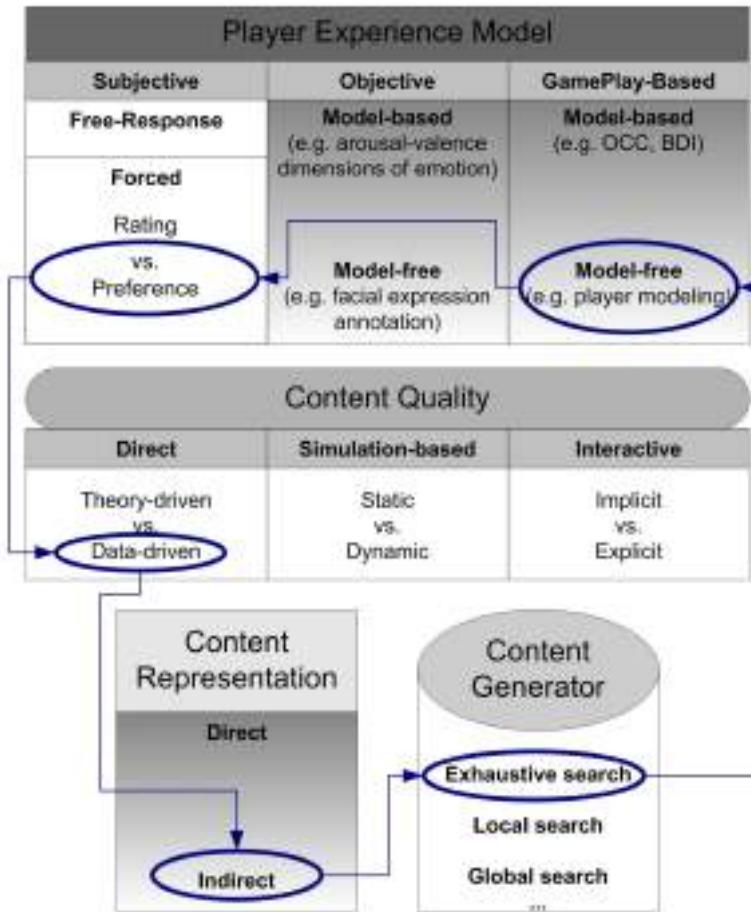


**Fig. 4.15** Example levels generated for two different Mario players. The generated levels maximize the modeled *fun* value for each player. The level on top is generated for one of the experiment subjects that participated in [521] while the level below is generated for the world champion agent of the Mario AI competition in 2009.

popular *The Legend of Zelda* (Nintendo, 1986–2017) action-RPG game series, is built mainly to support experience-driven PCG research. Another example is the work by Pedersen et al. [521], who modified an open-source clone of the classic platform game *Super Mario Bros* (Nintendo, 1985) to allow for personalized level generation. The realization of EDPCG in this example is illustrated in Fig. 4.16. The first step was to represent the levels in a format that would yield an easily searchable space. A level was represented as a short parameter vector describing the number, size and placement of gaps which the player can fall through, and the presence or absence of a switching mechanic. The next step was to create a model of player experience based on the level played and the player’s playing style. Data was collected from hundreds of players, who played pairs of levels with different parameters and were asked to rank which of two levels best induced each of the following user states: fun, challenge, frustration, predictability, anxiety, boredom. While playing, the game also recorded a number of metrics of the players’ playing styles, such as the frequency of jumping, running and shooting. This data was then used to train neural networks to predict the examined user states using evolutionary preference learning. Finally, these player experience models were utilized to optimize game levels for particular players [617]. Two examples of such levels can be seen in Fig. 4.15. It is worth noting—as discussed in Chapter 5—that one may wish to further improve the models of experience of Mario players by including information about the player beyond gameplay [29] such as her head pose [610] or her facial expressions [52].

#### 4.4.4 Experience-Agnostic

With experience-agnostic PCG we refer to any PCG approach that does not consider the role of the player in the generation of content. But where should we set the boundary of involvement? When do we consider a player as part of the generation process and when don’t we? While the borderline between experience-driven and experience-agnostic is not trivial to draw we define any PCG approach whose content quality function does not include a player (experience) model or it does not



**Fig. 4.16** The EDPCG framework in detail. The gradient grayscale-colored boxes represent a continuum of possibilities between the two ends of the box while white boxes represent discrete, exclusive options within the box. The blue arrows illustrate the EDPCG approach followed for the *Super Mario Bros* (Nintendo, 1985) example study [521, 617]: Content quality is assessed via a direct, data-driven evaluation function which is based on a combination of a gameplay-based (model-free) and a subjective (pairwise preference) player experience modeling approach; content is represented indirectly and exhaustive search is applied to generate better content.

interact with the player in any way during generation as experience-agnostic. As with the role of autonomous PCG, this chapter has already gone through several examples of content generation that do not involve a player or a player experience model. To avoid being repetitive we will refer the reader to the PCG studies covered already that are outside the definition of experience-driven PCG.

## 4.5 What Could Be Generated?

In this section we briefly outline the possible content types that a PCG algorithm can generate in a game. Generally speaking Liapis et al. [381] identified six creative domains (or else facets) within games that we will follow for our discussion in this section. These include level architecture (design), audio, visuals, rules (game design), narrative, and gameplay. In this chapter we will cover the first five facets and we purposely exclude the gameplay facet. Creative gameplay is directly associated with play and as such is covered in the previous chapter. We conclude this section with a discussion on complete game generation.

### 4.5.1 Levels and Maps

The generation of levels is by far the most popular use of PCG in games. Levels can be viewed as **necessary** content since every game has some form of spatial representation or virtual world within which the player can perform a set of actions. The properties of the game level, in conjunction with the game rules, frame the ways a player can interact with the world and determine how the player can progress from one point in the game to another. The game's level design contributes to the challenges a player faces during the game. While games would often have a fixed set of mechanics throughout, the way a level is designed can influence the gameplay and the degree of game challenge. For that reason, a number of researchers have argued that levels coupled with game rules define the absolutely necessary building blocks of any game; in that regard the remaining facets covered below are optional [371]. The variations of possible level designs are endless: a level representation can vary from simple two-dimensional illustrations of platforms and coins—as in the *Super Mario Bros* (Nintendo, 1985) series—to the constrained 2D space of *Candy Crush Saga* (King, 2012), to the three-dimensional and large urban spaces of *Assassin's Creed* (Ubisoft, 2007) and *Call of Duty* (Infinity Ward, 2003), to the 2D elaborated structures of *Angry Birds* (Rovio, 2009), to the voxel-based open gameworld of *Minecraft* (Mojang 2011).

Due to their several similarities we can view the procedural generation of game levels from the lens of procedural architecture. Similarly to architecture, level design needs to consider both the **aesthetic** properties and the **functional** requirements of whatever is designed within the game world. Depending on the game genre, functional requirements may vary from a reachable end-goal for platform games, to a challenging gameplay in driving games such as *Forza Motorsport* (Turn 10 Studios 2005), to waves of gameplay intensity as in *Pac-Man* (Namco, 1980), *Left 4 Dead* (Valve, 2008), *Resident Evil 4* (Capcom, 2005) and several other games. A procedural level generator also needs to consider the aesthetics of the content as the level's aesthetic appearance may have a significant impact not only on the visual stimuli it offers to the player but also on navigation. For example, a sequence of identical rooms can easily make the player disoriented—as was intended in the



**Fig. 4.17** The procedurally generated levels of *Diablo* (Blizzard Entertainment, 1996): one of the most characteristic examples of level generation in commercial games. *Diablo* is a relatively recent descendant of *Rogue* (Toy and Wichmann, 1980) (i.e., rogue-like) role-playing game characterized by dungeon-based procedurally generated game levels. Image obtained from Wikipedia (fair use).

dream sequences of *Max Payne* (Remedy, 2001)—while dark areas can add to the challenge of the game due to low visibility or augment the player’s arousal as in the case of *Amnesia: The Dark Descent* (Frictional Games, 2010), *Nevermind* (Flying Mollusk, 2015) and *Sonancia* [394]. When the level generator considers larger, open levels or gameworlds then it draws inspiration from urban and city planning [410], with edges to constrain player freedom—landmarks to orient the player and motivate exploration [381]—as in the *Grand Theft Auto* (Rockstar Games, 1997) series and districts and areas to break the world’s monotony—as in *World of Warcraft* (Blizzard Entertainment, 2004) which uses highly contrasting colors, vegetation and architectural styles to split the world into districts that are suitable for characters of different level ranges.

As already seen broadly in this chapter the generation of levels in a procedural manner is clearly the most popular and possibly the oldest form of PCG in the game industry. We already mentioned the early commercial use of PCG for automatic level design in games such as *Rogue* (Toy and Wichman, 1980) and the Rogue-inspired *Diablo* (see Fig. 4.17) series (Blizzard Entertainment, 1996), and the more recent world generation examples of *Civilization IV* (Firaxis, 2005) and *Minecraft* (Mojang, 2011). The level generation algorithms used in commercial games are usually



**Fig. 4.18** The caricaturized and highly-emotive visuals of the game *Limbo* (Playdead, 2010). Image obtained from Wikipedia (fair use).

constructive, in particular, in games where players can interact with and change the game world via play. Players of *Spelunky* (Yu and Hull, 2009), for instance, are allowed to modify a level which is not playable (i.e., the exit cannot be reached) by blowing up the blocking tiles with a bomb provided by the game level.

The academic interest in procedural level generation is only recent [720, 783, 616] but it has produced a substantial volume of studies already. Most of the academic studies described in this chapter are focused on levels. The interested reader may refer to those for examples of level generation across various methodologies, PCG roles and game genres.

### 4.5.2 Visuals

Games are, by definition, visual media unless the game is designed explicitly to not have visuals—e.g., the *Real Sound: Kaze no Regret* (Sega, 1997) adventure audio game. The visual information displayed on the screen conveys messages to the player which are dependent on the graphical style, color palette and visual texture. Visuals in games can vary from simple abstract and pixelized representations as the 8-bit art of early arcade games, to caricaturized visuals as in *Limbo* (Playdead, 2010) (see Fig. 4.18), to photorealistic graphics as in the *FIFA* series (EA Sports, 1993) [299].

Within the game industry PCG has been used broadly for the generation of any of the above visual types. Arguably, the complexity of the visual generation task increases the more the resolution and the photorealism of the desired output increases. There are examples of popular generative tools such as *SpeedTree* (IDV, 2002) for vegetation and *FaceGen* (Singular Inversions, 2001) for faces, however, that can successfully output photorealistic 3D visuals. Within academia notable examples of visual generation are the games *Petalz* [565, 566] (flower generation), *Galactic Arms Race* [250] (weapon generation; see also Fig. 4.3) and *AudioInSpace*

[275] (weapon generation). In all three games visuals are represented by neural networks that are evolved via interactive evolution. Within the domain of weapon particle generation another notable example is the generation of surprising yet balanced weapons for the game *Unreal Tournament III* (Midway Games, 2007) using **constrained surprise search** [240]; an algorithm that maximizes the surprise score of a weapon but at the same time imposes designer constraints to it so that it is balanced. Other researchers have been inspired by theories about “universal” properties of beauty [18] to generate visuals of high appeal and appreciation [377]. The PCG algorithm in that study generates spaceships based on their size, simplicity, balance and symmetry, and adapts its visual output to the taste of the visual’s designer via interactive evolution. The PCG-assisted design process referred to as *iterative refinement* [380] is another way of gradually increasing the resolution of the visuals a designer creates by being engaged in an iterative and creative dialog with the visuals generator. Beyond the generation of in-game entities a visuals PCG algorithm may focus on general properties of the visual output such as pixel shaders [282], lighting [188, 185], brightness and saturation, which can all influence the overall appearance of any game scene.

### 4.5.3 Audio

Even though audio can be seen as optional content it can affect the player directly and its impact on player experience is apparent in most games [219, 221, 129]. Audio in games has reached a great level of maturity as demonstrated by two BAFTA Game Awards and an MTV video music award for best video game soundtrack [381]. The audio types one meets in games may vary from fully orchestrated soundtrack (background) music, as in *Skyrim* (Bethesda, 2011), to sound effects, as the dying or pellet-eating sounds of *Pac-Man* (Namco, 1980), to the voice-acted sounds of *Fallout 3* (Bethesda, 2008). Most notably within indie game development, *Proteus* (Key and Kanaga, 2013) features a mapping between spatial positioning, visuals and player interaction, which collectively affect the sounds that are played. Professional tools such as the sound middleware of UDK (Epic Games, 2004) and the popular *sfxr* and *bfxr* sound generation tools provide procedural sound components to audio designers, demonstrating a commercial interest in and need of procedurally generated audio.

At a first glance, the generation of game audio, music and sounds might not seem to be particularly different from any other type of audio generation outside games. Games are interactive, however, and that particular feature makes the generation of audio a rather challenging task. When it comes to the definition of procedural audio in games, a progressive stance has been that its procedurality is caused by the very interaction with the game. (For instance, game actions that cause sound effects or music can be considered as procedural audio creation [220, 128].) Ideally, the game audio must be able to adapt to the current game state and the player behavior. As a result adaptive audio is a grand challenge for composers since the combinations of

all possible game and player states could be largely unknown. Another difficulty for the autonomous generation of adaptive music, in particular, is that it requires real-time composition and production; both of which need to be embedded in a game engine. Aside from a few efforts in that direction<sup>3</sup> the current music generation models are not particularly designed to perform well in games. In the last decade, however, academic work on procedural game music and sound has seen substantial advancements in venues such as the Procedural Content Generation and the Musical Metacreation workshop series.

Generally speaking, sound can be either **diegetic** or **non-diegetic**. A sound is diegetic if its source is within the game's world. The source of the diegetic sound can be visible on the screen (on-screen) or can be implied to be present at the current game state (off-screen). Diegetic sounds include characters, voices, sounds made by items on-screen or interactions of the player, or even music represented as coming from instruments available in the game. A non-diegetic sound, on the other hand, is any sound whose source is outside the game's world. As such non-diegetic sounds cannot be visible on-screen or even implied to be off-screen. Examples include commentaries of a narrator, sound effects which are not linked to game actions, and background music.

A PCG algorithm can generate both diegetic and non-diegetic sounds including music, sound effects and commentaries. Examples of non-diegetic sound generation in games include the *Sonancia* horror sound generator that tailors the tension of the game to the desires of the designer based on the properties of the game level [394]. The mapping between tension and sounds in *Sonancia* has been derived through crowdsourcing [396]. Similarly to *Sonancia*—and towards exploring the creative space between audio and level design—*Audiooverdrive* generates levels from audio and audio from levels [273]. Notably within diegetic audio examples, Scirea et al. [606] explores the relationship between procedurally generated music and narrative. Studies have also considered the presence of game characters on-display for the composition of game soundtracks [73] or the combination of short musical phrases that are driven by in-game events and, in turn, create responsive background audio for strategy games [280].

Finally, it is worth mentioning that there are games featuring PCG that use music as the input for the generation of other creative domains rather than music per se. For example, games such as *Audio Surf* (Fitterer, 2008) and *Vib Ribbon* (Sony Entertainment, 2000) do not procedurally generate music but they instead use music to drive the generation of levels. *AudioInSpace* [275] is another example of a side-scrolling space shooter game that does not generate music but uses the background music as the basis for weapon particle generation via artificial evolution.

---

<sup>3</sup> For instance, see the upcoming melodrive app at: <http://melodrive.com>.

#### 4.5.4 Narrative

Many successful games are relying heavily on their narratives; the clear distinction however, between such narratives and traditional stories is the interactivity element that is offered by games. Now, whether games can tell stories [313] or games are instead a form of narrative [1] is still an open research question within game studies, and beyond the scope of this book. The study of computational (or procedural) narrative focuses on the representational and generational aspects of stories as those can be told via a game. Stories can play an essential part in creating the aesthetics of a game which, in turn, can impact affective and cognitive aspects of the playing experience [510].

By breaking the game narrative into subareas of game content we can find core game content elements such as the game's plotline [562, 229], but also the ways a story is represented in the game environment [730, 83]. The coupling of a game's representation and the **story of the game** is of vital importance for player experience. Stories and plots are taking place in an environment and are usually told via a **virtual camera** lens. The behavior of the virtual camera—viewed as a parameterized element of computational narrative—can drastically influence the player's experience. That can be achieved via an affect-based cinematographic representation of multiple cameras as those used in *Heavy Rain* (Quantic Dream, 2010) or through an affect-based automatic camera controller as that used in the Maze-Ball game [780]. Choosing the best camera angle to highlight an aspect of a story can be seen as a multi-level optimization problem, and approached with combinations of optimization algorithms [85]. Games such as *World of Warcraft* (Blizzard Entertainment, 2004) use cut scenes to raise the story's climax and lead the player to particular player experience states. The creation or semi-automatic generation of stories and narratives belongs to the area of interactive storytelling, which can be viewed as a form of story-based PCG. The story can adjust according to the actions of the player targeting personalized story generation (e.g., see [568, 106] among others). Ultimately, game worlds and plot point story representations can be co-generated as demonstrated in a few recent studies (e.g., see [248]).

Computational narrative methods for generating or adapting stories of expositions are typically build on planning algorithms, and **planning** is therefore essential for narrative [792]. The space of stories can be represented in various ways, and the representations in turn make use of dissimilar search/planning algorithms, including traditional optimization and reinforcement learning approaches [483, 117, 106]. Cavazza et al. [106], for instance, introduced an interactive storytelling system built with the Unreal game engine that uses Hierarchical Task Network planning to support story generation and anytime user intervention. Young et al. [792] introduced an architecture called *Mimesis*, primarily designed to generate intelligent, plan-based character and system behavior at runtime with direct uses in narrative generation. Finally the IDtension engine [682] dynamically generates story paths based on the

player's choices; the engine was featured in *Nothing for Dinner*,<sup>4</sup> a 3D interactive story aiming to help teenagers living challenging daily life situations at home.

Similarly to dominant approaches of narrative and text generation, interactive storytelling in games relies heavily on stored knowledge about the (game) world. Games that rely on narratives—such as *Heavy Rain* (Quantic Dream, 2010)—may include thousands of lines of dialog which are manually authored by several writers. To enable interactive storytelling the game should be able to select responses (or paths in the narrative) based on what the player will do or say, as in *Façade* [441] (see Fig. 4.19). To alleviate, in part, the burden of manually representing world knowledge, data-driven approaches can be used. For instance, one may crowdsource actions and utterance data from thousand of players that interact with virtual agents of a game and then train virtual agents to respond in similar ways using *n*-grams [508]. Or instead, one might design a system in which designers collaborate with a computer by taking turns on adding sentences in a story; the computer is able to provide meaningful sentences by matching the current story with similar stories available on the cloud [673]. Alternatively, a designer could use the news of the day from sites, blogs or Wikipedia and generate games that tell the news implicitly via play [137].

Research on interactive narrative and story-based PCG benefits from and influences the use of **believable agents** that interact with the player and are interwoven in the story plot. The narrative can yield more (or less) believability to agents and thus the relationship between agents and the story they tell is important [801, 401, 531, 106]. In that sense, the computational narrative of a game may define the arena for believable agent design. Research on story-based PCG has also influenced the design and development of games. Starting from popular independent attempts like *Façade* [441] (see Fig. 4.19), *Prom Week* [448] and *Nothing for Dinner* to the commercial success of *The Elder Scrolls V: Skyrim* (Bethesda Softworks, 2011), *Heavy Rain* (Quantic Dream, 2010) and *Mass Effect* (Bioware, 2007) narrative has traditionally been amongst the key factors of player experience and immersion; particularly in narrative-heavy games as the ones aforementioned.

Examples of sophisticated computational narrative techniques crossing over from academia to commercial-standard products include the storytelling system *Versu* [197] which was used to produce the game *Blood & Laurels* (Emily Short, 2014). For the interested reader the *interactive fiction database*<sup>5</sup> contains a detailed list of games built on the principles of interactive narratives and fiction, and the stoygen.org<sup>6</sup> repository, by Chris Martens and Rogelio E. Cardona-Rivera, maintains existing openly-available computational story generation systems. Finally note that the various ways AI can be used to play text-based adventure games and interactive fiction are covered in Chapter 3.

---

<sup>4</sup> <http://nothingfordinner.org>

<sup>5</sup> <http://ifdb.tads.org/>

<sup>6</sup> <http://stoygen.org/>



**Fig. 4.19** A screenshot from the *Façade* [441] game featuring its main characters: Grace and Trip. The seminal design and AI technologies of *Façade* popularized the vision of interactive narrative and story-based PCG within games.

#### 4.5.5 Rules and Mechanics

The game rules frame the playing experience by providing the conditions of play—for instance, winning and losing conditions—and the actions available to the player (game mechanics). Rules constitute necessary content as they are in a sense the core of any game, and a game's rules pervade it.

For most games, the design of their ruleset largely defines them and contributes to their success. It is common that the rule set follows some standard design patterns within its genre. For example, the genre of platform games is partly defined by running and jumping mechanics, whereas these are rare in puzzle games. Evidently, the genre constrains the possibility (design) space of the game rules. While this practice has been beneficial—as rule sets are built on earlier successful paradigms—it can also be detrimental to the creativity of the designer. It is often the case that the players themselves can create new successful game variants (or even sub-genres) by merely altering some rules of an existing game. A popular example is the modification of *Warcraft III* (Blizzard, 2002) which allowed the player to control a single “hero” unit and, in turn, gave rise to a new, popular game genre named Multiplayer Online Battle Arenas (MOBA).

Most existing approaches to rule generation take a search-based approach, and are thus dependent on some way of evaluating a set of rules [711, 355]. However, accurately estimating the quality of a set of game rules is very hard. Game rules differ from most other types of game content in that they are almost impossible to evaluate in isolation from the rest of the game. While levels, characters, textures, and many other types of content can to some extent be evaluated outside of the game, looking at a set of rules generally gives very little information of how they play. For a human, the only way to truly experience the rules of a game is to play the game. For a computer, this would translate to simulating gameplay in some way in order to evaluate the rules. (In this sense, rules can be said to be more similar to program code than they are to e.g., pictures or music.)

So how can simulated playthroughs be used to judge the quality of the rulesets? Several ideas about how to judge a game depending on how agents play it have been introduced. The first is **balance**; for symmetric two-player games in particular, balance between the winning chances of the two players is generally positive [274]. Another idea is **outcome uncertainty**, meaning that any particular game should be “decided” as late as possible [76]. Yet another idea is **learnability**: a good game, including its ruleset, is easy to learn and hard to master. In other words, it should have a long, smooth learning curve for the player, as learning to play the game is part of what makes it fun. This idea can be found expressed most clearly in Koster’s “Theory of Fun” [351], but can also be said to be implicit in Schmidhuber’s theory of artificial curiosity [602] and in theories in developmental psychology [204]. Within work in game rule generation, attempts have been made to capture this idea in different ways. One way is to use a reinforcement learning agent to try to learn to play the game; games where the agent improves the most over a long time score the best [716]. Another way of capturing this idea is to use several agents of different skill levels to try to play the game. Games score better when they maximize the performance difference between these agents [491]. This idea is also related to the idea of **depth** in games, which can be seen as the length of the chain of heuristics that can be acquired in a game [362].

Perhaps the most successful example of game rule generation within academic research is *Ludi* [76]. *Ludi* follows a search-based PCG approach and evolves grammars that represent the rules of board games (see Fig. 4.20). The fitness function that drives the rule generation is composed by several metrics that estimate good design patterns in board games, such as game depth and rule simplicity. A successfully designed game that came out of the *Ludi* generator is named *Yavalath* [75] (see Fig. 4.20). The game is played on a 5-by-5 hexagonal board by two or three players. *Yavalath*’s winning and losing conditions are very simple: you win if you make a line of four tokens, whereas you lose if you make a line of three tokens; the game is a draw if the board fills up.

One of the earliest examples of *video game* rule generation is Togelius and Schmidhuber’s experiment with generating simple Pac-Man-like games [716]. In that study rules are evolved to maximize the learnability of player agents as measured via simulated playthroughs. Another example is the work by Nielsen et al. in which game rules are represented using the video game description language [492].

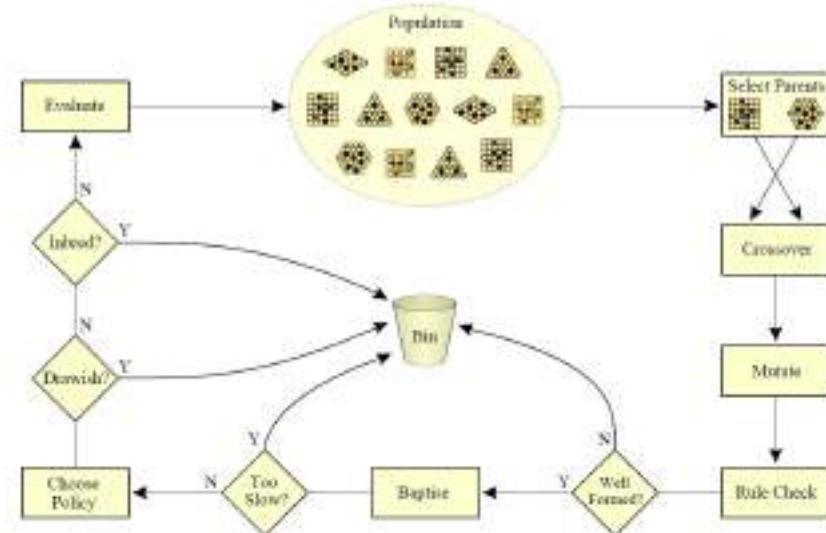
Using Answer Set Programming (a solver-based method) [69] rather than search-based methods, rules have been generated for simple 2D games that, however, respect constraints such as playability (i.e., the victory condition is attainable) [637]. It is fair to say that none of these attempts to generate video games have been able to produce *good* games, i.e., games that anyone (except the creator of the system that creates the games) would want to play. This points to the immense challenge in accurately estimating the quality of a video game rule set. One of the reasons this seems to be a more challenging problem than estimating the quality of a board game rule set is the time dimension, as human reaction time and ability to estimate and predict unfolding processes play an important role in the challenge of many video games.

For more examples and in-depth analysis on rule and mechanics generation the interested reader is referred to the “rules and mechanics” chapter of the PCG book [486].

#### 4.5.6 Games

**Game generation** refers to the use of PCG algorithms for computationally designing new complete games. The vast majority of PCG studies so far, however, have been very *specific* to a particular game facet or domain. It is, for instance, either a level that is generated or the audio for some level but rarely both. Meanwhile it is surprising to think that the relationship between the different facets is naturally interwoven. A characteristic example of the interwoven nature among game facets is given in [381]: player actions—viewed as a manifestation of game rules—are usually accompanied by corresponding sound effects such as the sound of Mario jumping in *Super Mario Bros* (Nintendo, 1985). Now let us think of a PCG algorithm that introduces a new rule to the game—hence a new player action. The algorithm automatically constrains the sound effects that can be associated to this new action based on a number of factors such as the action’s duration, purpose and overall contribution to the game plot. Actions and sounds appear to have a cause and effect (or hierarchical) relationship and a PCG algorithm would naturally prioritize the creation of the action before it generates its sound. Most relationships between facets, however, are not strictly hierarchical or unidirectional. For example, a game level can be successful because of a memorable landmark as much as the gameplay it affords [381]. Similarly, the narrative of a game relies on a multitude of factors including the camera placement as well as the visuals and the sounds.

The game generation topic has attracted a growing interest in recent years even though the relationship between the different games facets is not considered largely. Most game generation projects focus on a single facet of a game and do not investigate the interaction between different facets. The rule generator for Pac-Man-like games [716], for instance, evolves different rules for different colored agents but it does not evolve the agents’ color to indicate different playing strategies. Similarly

(a) The *Ludi* game rule generator(b) A deluxe version of the *Yavalath* game

**Fig. 4.20** The *Ludi* game rule generator (a) and its game *Yavalath* (b). Image (a) is adapted from [720]. Image (b) is published with permission by Cameron Browne and Néstor Romeral Andrés.

to the ghosts of *Pac-Man* (Namco, 1981) we could imagine that red represents an aggressive behavior whereas orange represents a passive behavior.

Among the few notable attempts of game generation, Game-O-Matic [723] is a game generation system that creates games representing ideas. More specifically, Game-O-Matic includes an authoring tool in which a user can enter entities and their interactions via concept maps. The entities and interactions are translated, respectively, into game visuals and game mechanics; the mechanics, however, do not take into account the visuals or semantics of the game objects they are applied on. One of the first preliminary discussions on how multi-facet integration might happen is offered by Nelson and Mateas [484]. In their paper they present a system for matching sprites (visuals) to very simple WarioWare-style mechanics. Their system is somewhat similar to Game-O-Matic, but works a bit differently: instead of the designer specifying verbs and nouns she wants a game to be about, she gives the system constraints on how verbs and nouns relate in the game (for example, a chasing game needs a “prey” sprite that is something that can do things like “flee” or “be hunted” and so on). The system then uses ConceptNet<sup>7</sup> and WordNet<sup>8</sup> to generate games that fit these constraints.

Arguably one of the most elaborate examples of game generation is ANGELINA [135, 137, 136]. ANGELINA<sup>9</sup> is a game generator that has seen several developments over the years and is currently capable of evolving the rules and levels of the game, collecting and placing pieces of visuals and music (that are relevant to the theme and the emotive mood of the game), giving names for the games it creates and even creating simple commentaries that characterize them. ANGELINA is able to generate games of different genres—including platformer games (see Fig. 4.21) and 3D adventure games—some of which have even participated in game design competitions [136].

The systems above make some initial, yet important, steps towards game generation and they attempt to interweave the different domains within games in a meaningful way, mostly in a hierarchical fashion. However, PCG eventually needs to rise to the challenge of tackling the compound generation of multiple facets in an **orchestrated** manner [711, 371]. An early study on the fusion of more than one generative facet (domain) in games is the one performed recently by Karavolos et al. [324]. The study employs machine learning-based PCG to derive the common generative space—or the common patterns—of game levels and weapons in first-person shooters. The aim of this orchestration process between level design and game design is the generation of level-weapon couplings that are balanced. The unknown mapping between level representations and weapon parameters is learned by a deep convolutional neural network which predicts if a given level with a particular set of weapons will be balanced or not. Balance is measured in terms of the win-lose ratio obtained by AI bots playing in a deathmatch scenario. Figure 4.22 illustrates the architecture used to fuse the two domains. For the interested reader in the or-

---

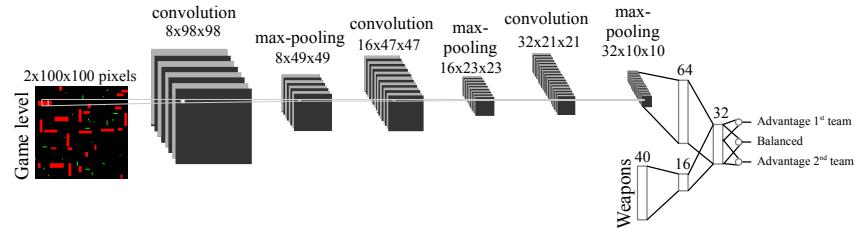
<sup>7</sup> <http://conceptnet.io/>

<sup>8</sup> <https://wordnet.princeton.edu/>

<sup>9</sup> <http://www.gamesbyangelina.org/>



**Fig. 4.21** ANGELINA’s Puzzling Present game. The game features an invert gravity mechanic that allows a player to overcome the high obstacle on the left and complete the level. Image obtained with permission from <http://www.gamesbyangelina.org/>.



**Fig. 4.22** A convolutional neural network (CNN) architecture used for fusing levels and weapons in first-person shooters. The network is trained to predict whether a combination of a level and a weapon would yield a balanced game or not. The CNN can be used to orchestrate the generation of a balanced level given a particular weapon and vice versa. Image adapted from [324].

chestration process we further elaborate on the topic in the last chapter of this book; some early discussions on this vision can also be found in [371].

## 4.6 Evaluating Content Generators

Creating a generator is one thing; evaluating it is another. Regardless of the method followed all generators shall be evaluated on their ability to achieve the desired goals of the designer. Arguably, the generation of any content is trivial; the generation of **valuable** content for the task at hand, on the other hand, is a rather challenging procedure. (One may claim, however, that the very process of generating valuable content is also, by itself, trivial as one can design a generator that returns a random sample of hand-crafted masterpieces.) Further, it is more challenging to generate content that is not only valuable but is also novel or even inspiring.

### 4.6.1 Why Is It Difficult?

But what makes the evaluation of content so difficult? First, it is the diverse, stochastic and subjective nature of the **users** that experience the content. Whether players or designers, content users have dissimilar personalities, gameplay aims and behavior, emotive responses, intents, styles and goals [378]. When designing a PCG system it is critical to remember that we can potentially generate massive amounts of content for designers to interact with and players to experience. It is thus of utmost importance to be able to evaluate how successful the outcomes of the generator might be across dissimilar users: players and designers. While content generation is a cheap process relying on algorithms, design and game-play are expensive tasks relying on humans who cannot afford the experience of bad content. Second, content quality might be affected by **algorithms** and their underlying stochasticity, for instance, in evolutionary search. Content generators often exhibit non-deterministic behavior, making it very hard to predict *a priori* what the outcomes of a particular generative system might be.

### 4.6.2 Function vs. Aesthetics

Particular properties of content can be **objectively** defined and tested whereas other properties of it can only be assessed **subjectively**. It is only natural to expect that functional properties of content quality can be objectively defined whereas a large part of its aesthetics can only be defined subjectively. For instance, playability of a level is a functional characteristic that can be objectively measured—e.g., an AI agent manages to complete the level; hence it is playable. Balance and symmetry can also be objectively defined to a degree through estimates of deviation from a norm—it may be a score (balance) or a distance from a central choke point in the map (symmetry). There are games, however, for which content balance, symmetry and other functional properties are not trivially measurable. And of course there are several aspects of content such as the comprehensibility of a narrative, the pleas-

antnesses of a color scheme, the preference for a room's architectural style or the graphics style, and the experience of sound effects and music that are not objectively measured.

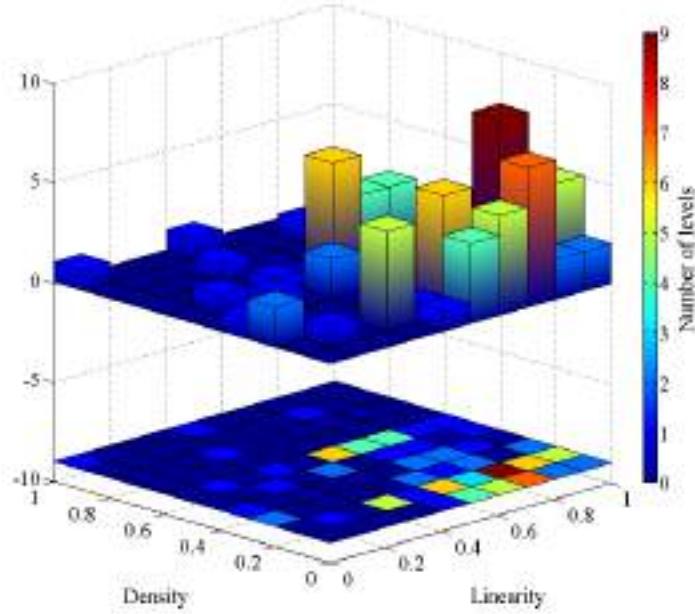
Functional, objectively defined, content properties can be expressed either as metrics or as constraints that a generator needs to satisfy. **Constraints** can be specified by the content designers or imposed by other game content already available. For instance, let us assume that a well-designed generated strategy game level needs to be both balanced and playable. Playability can form a simple binary constraint: the level is playable when an AI agent manages to complete it; it is non-playable otherwise. Balance can form another constraint by which all items, bases and resources are accessible to similar degrees by all players; if equal accessibility is below a threshold value then the constraint is not satisfied. Next, let us suppose we wish to generate a new puzzle for the map we just generated. Naturally, the puzzle needs to be compatible with our level. A PCG algorithm needs to be able to satisfy these constraints as part of its quality evaluation. Constrained satisfaction algorithms such as the feasible-infeasible two-population genetic algorithm [379, 382], constrained divergent search rewarding content *value* but also content *novelty* [382] or *surprise* [240], and constraint solvers such as answer set programming [638] are able to handle this. The generated results are within constraints, thereby valuable for the designer. Value, however, may have varying degrees of success and this is where alternative methods or heuristics can help, as we cover in the section below.

### 4.6.3 How Can We Evaluate a Generator?

Generally speaking, a content generator can be evaluated in three ways: directly by the **designer** or indirectly by either **human players** or **AI agents**. Designers can directly observe properties of the content generator and take decisions based on data **visualization** methods. Human players can play and test the content and/or provide feedback about the content via subjective reporting. AI agents can do the same: play the content or measure something about the content and report it to us in the form of a quality metric, or metrics. Clearly, machines cannot *experience* the content but they can, instead, simulate it and provide us estimates of content experience. The overall evaluation process can very well combine and benefit from any of the above approaches. In the remainder of this section we cover the approaches of data visualization, AI automated playtesting and human playtesting in further detail.

#### 4.6.3.1 Visualization

The visualization approach to content quality assessment is associated with a) the computation of meaningful metrics that can assign measurable characteristics to content and b) ways to visualize these metrics. The task of metric design can be viewed as the equivalent of fitness function design. As such, designing good con-



**Fig. 4.23** The expressive range of the *Ropossum* level generator for the metrics of linearity and density. Adapted from [608].

tent generation quality metrics in an ad-hoc manner involves a degree of practical wisdom. Metrics, for instance, can be based on the **expressive range** of the generator under assessment, so-called expressivity metrics [640, 608]. The analysis of a generator's expressivity gives us access to the potential overall quality of the generator across its full range of generative space. The generative space can then be visualized as **heatmaps** or alternative graphical representations such as 2D or 3D scatter plots (see Fig. 4.23 for an example). It is one thing if a level generator is able to create only a few meaningful or playable levels and another if the generator is robust and consistent with respect to the playability of its generated levels. It is also one thing if our generator is only able to generate levels with very specific characteristics within a narrow space of its expressive range and another if our level generator is able to express a broad spectrum of level properties, yielding uniformly covered expressive ranges. Such information can be directly visible on the illustrated heatmaps or scatter plots. Alternatively, **data compression** methods can be used directly on the generated content and offer us 2D or 3D representations of the generative space, thereby, bypassing the limitations of ad-hoc metric design. An example of this approach is the use of autoencoders for compressing the images produced by the DeLeNoX autonomous content generator [373].

#### 4.6.3.2 AI

Using AI to **playtest** our generator is a safe and relatively cheap way to rapidly retrieve quality metrics for it without relying on human playtesting. In the same way that a search-based PCG method would use AI to simulate the content before generating it, AI agents can test the potential of a generator across a number of metrics and return us values about its quality. The metrics can be in the form of classes—for instance, test checks performed for completing an area of a level—scalar values—e.g., a level’s balance—or even ordinal values—e.g., the rank of the level in terms of asymmetry. The relevance of the metrics to the generator’s quality is obviously dependent on the designer’s ad-hoc decisions. Once again, designing appropriate metrics for our AI agent to compute is comparable to the challenge of designing any utility function. An interesting approach to AI-based testing is the use of **procedural personas** [267, 269]. These are data-driven inferred models of dissimilar play styles that potentially imitate the different styles of human play. In a sense, procedural personas provide a more human-realistic approach to AI-based testing of a generator. Finally, by visualizing particular game artifacts or simulating them through the use of AI agents we can have access to the information we might be able to extract from a game, we can understand what is possible within our content space, we can infer how rules and functions operate in whatever we generate, and we can possibly understand how the information we are able to extract relates to data we can extract from human playtesting [481].

#### 4.6.3.3 Human Players

In addition to data visualization and AI-based simulation for the evaluation of a content generator a designer might wish to use complementary approaches that rely on quantitative user studies and playtesting. Playtesting is regarded to be an expensive way to test a generator but it can be of immense benefit for content quality assurance, in particular for those aspects of content that cannot be measured objectively—e.g., aesthetics and playing experience. The most obvious approach to evaluate the content experienced by players is to *explicitly* ask them about it. A game user study can involve a small number of dedicated players that will play through various amounts of content or, alternatively, a crowdsourcing approach can provide sufficient data to machine learn content evaluation functions (see [621, 370, 121] among others). Data obtained can be in any format including classes (e.g., a binary answer about the quality of a level), scores (e.g., the likeliness of a sound) or ranks (e.g., a preference about a particular level). It is important to note that the playtesting of content can be complemented by annotations coming from the designers of the game or other experts involved in content creation. In other words, our content generator may be labeled with both first-person (player) and third-person (designer) annotations. Further guidelines about which questionnaire type to use and advice about the design of user study protocols can be found in Chapter 5.

## 4.7 Further Reading

Extensive versions of most of the material covered in this chapter can be found in dedicated chapters of the PCG in Games Textbook [616]. In particular, all the methods and types of PCG in games are covered in further detail (see Chapters 1 to 9 of [616]). Considering the roles of PCG, the mixed-initiative and the experience-driven role of PCG are, respectively, detailed in Chapters 11 [374] and 10 [618] of that textbook [616]. In addition to Chapter 11 of [616] the framework named *mixed-initiative co-creativity* [774] provides a theoretical grounding for the impact of mixed-initiative interaction on the creativity of both designers and computational processes. Further, the original articles about the experience-driven PCG framework can be found in [783, 784]. Finally, the topic of PCG evaluation is covered also in the final chapter [615] of the PCG in Games textbook [616].

## 4.8 Exercises

PCG offers endless opportunities for generation and evaluation across the different creativity domains in games and across combinations of those. As an initial step we would recommend the reader to start experimenting with maze generation and platformer level generation (as outlined below). The website of the book contains details regarding both frameworks and potential exercises.

### 4.8.1 Maze Generation

Maze generation is a very popular type of level generation and relevant for several game genres. In the first exercise we recommend that you develop a maze generator using both a constructive and a search-based PCG approach and compare their performance according to a number of meaningful criteria that you will define. The reader may use the Unity 3D open-access maze generation framework which is available at: <http://catlikecoding.com/unity/tutorials/maze/>. Further guidelines and exercises for maze generation can be found at the book's website.

### 4.8.2 Platformer Level Generation

The platformer level generation framework is based on the *Infinite Mario Bros* (Persson, 2008) framework which has been used as the main framework of the Mario AI (and later Platformer AI) Competition since 2010. The competition featured several different tracks including gameplay, learning, Turing test and level generation. For the exercises of this chapter the reader is requested to download the level generation

framework (<https://sites.google.com/site/platformersai/>) and apply constructive and generate-and-test methods for the generation of platformer levels. The levels need to be evaluated using one or more of the methods covered in this book. Further details and exercises with the platformer level generation framework can be found at the book's website.

## 4.9 Summary

This chapter viewed AI as a means for generating content in games. We defined procedural content generation as the algorithmic process of creating content in and for games and we explored the various benefits of this process. We then provided a general taxonomy about content and its generation and explored the various ways one can generate content including search-based, solver-based, grammar-based, machine learning-based, and constructive generation methods. The use of the PCG method is naturally dependent on the task at hand and the type of content one wishes to generate. It further depends on the potential role the generator might take within games. We outlined the four possible roles a generator can take in games which are determined by the degree to which they involve the designer (autonomous vs. mixed-initiative) and/or the player (experience-agnostic vs. experience-driven) in the process. The chapter ends with a discussion on the important and rather unexplored topic of evaluation, the challenges it brings, and a number of evaluation approaches one might consider.

We have so far covered the most traditional use of AI in games (Chapter 3) and the use of AI for generating parts of (or complete) games (this chapter). The next and final chapter of this part is dedicated to the player and the ways we can use AI to model aspects of her behavior and her experience.

## Chapter 5

# Modeling Players

This chapter is dedicated to *players* and the use of AI for modeling them. This area of research is often called **player modeling** [782, 636]. We take player modeling to mean the detection, prediction and expression of human player characteristics that are manifested through cognitive, affective and behavioral patterns while playing games. In the context of this book, player modeling studies primarily the use of AI methods for the construction of computational models of players. By **model** we refer to a mathematical representation—it may be a rule set, a vector of parameters, or a set of probabilities—that captures the underlying function between the characteristics of the player and her interaction with the game, and the player’s response to that interaction. Given that every game features at least one player (with some notable exceptions [50]), and that player modeling affects work on game-playing and content generation, we consider the modeling of player behavior and experience as a very important use of AI in games [764, 785].

Psychology has studied human behavior, cognition and emotions for a long time. Branches of computer science and human-computer interaction that attempt to model and simulate human behavior, cognition, emotion or the feeling of emotion (**affect**) include the fields of **affective computing** and **user modeling**. Player modeling is related to these fields but focuses on the domain of games. Notably, games can yield dynamic and complex emotions in the player, the manifestations of which cannot be captured trivially by standard methods in empirical psychology, affective computing or cognitive modeling research. The high potential that games have in affecting players is mainly due to their ability to place the player in a continuous mode of interaction, which, in turn, elicits complex cognitive, affective and behavioral responses. Thus, the study of the player may not only contribute to the design of improved forms of human-computer interaction, but also advance our knowledge of human experiences.

As mentioned earlier, every game features at least one user—the player—who controls some aspect of the game environment. The player character could be visible in the game as an avatar or a group of entities [94], or could be invisible as in many puzzle games and casual games. Control may vary from the relatively simple (e.g., limited to movement in an orthogonal grid) to the highly complex (e.g., having

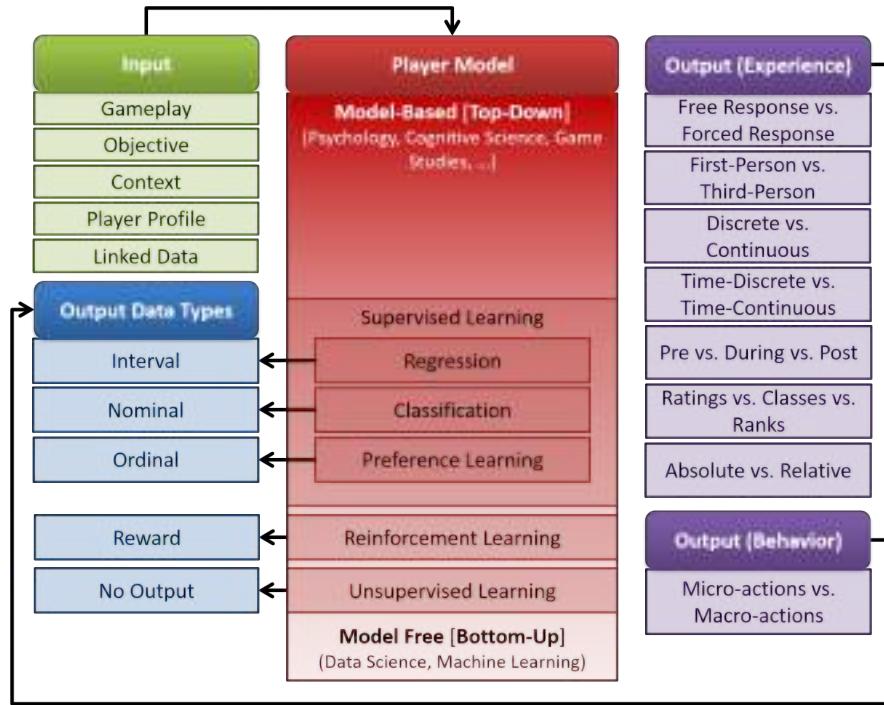
to decide several times per second between hundreds of different possibilities in a highly complex 3D world). Given these intricacies, understanding and modeling the interaction between the player and the game can be seen as a **holy grail** of game design and development. Designing the interaction and the emergent experience *right* results in a successful game that manages to elicit unique experiences.

The interaction between the player(s) and the game is dynamic, real-time and in many cases highly complex. The interaction is also **rich** in the sense that many modalities of interaction may be involved and that the information exchange between the game and the player may both be fast and entail large amounts of data for us to process. If the game is well-designed, the interaction is also highly engaging for the player. Given the great amount of information that can be extracted through this interaction and used for creating models of the player, the game should be able to learn much about the person playing it, as a player and perhaps as a human in general. In fact, there is no reason why the model should not know more about how you play than you do.

In the remainder of this chapter we first attempt to define the core ingredients of player modeling (Section 5.1) and then we discuss reasons why AI should be used to model players (Section 5.2). In Section 5.3 we provide a high-level taxonomy of player modeling focusing on two core approaches for constructing a player model: top-down and bottom-up. We then detail the available types of data for the model’s **input** (Section 5.4), a classification for the model’s **output** (Section 5.5) and the various AI methods that are appropriate for the player modeling task (Section 5.6). The key components of player modeling as discussed in this chapter (input, output and model) are depicted in Fig. 5.1. We conclude, in Section 5.7, with a number of concrete examples of AI being used for modeling players.

## 5.1 What Player Modeling Is and What It Is Not

One could arguably detect behavioral, emotional or cognitive aspects of both human players and non-human players, or non-player characters (notwithstanding the actual existence of emotions in the latter). However, in this book we focus on aspects that can be detected from, modeled from, and expressed in games with human players [782]. We explicitly exclude the modeling of NPCs from our discussion in this chapter, as in our definition, player modeling is modeling of a **human player**. Modeling the experience of an NPC would seem to be a futile exercise, as one can hardly say that an NPC possesses actual emotions or cognition. Modeling the behavior of an NPC is also of little interest, at least if one has access to the game’s code: a perfect model for the NPC already exists. NPC modeling, however, can be a useful testbed for player modeling techniques, for instance, by comparing the model derived from human players with the hand-crafted one. More interestingly, it can be an integral component of AI that adapts its behavior in response to the dynamics of the NPCs—as in [28]. Nevertheless, while the challenges faced in modeling NPCs



**Fig. 5.1** The key components of player modeling as discussed in this chapter. The distinction between model-based and model-free approaches is outlined in Section 5.3. The various options for the input of the model are discussed in Section 5.4. The taxonomy for the model’s output is discussed in Section 5.5—each box represents a dedicated subsection. Finally, the various AI methods (supervised learning, reinforcement learning and unsupervised learning) used for modeling corresponding output data types are discussed thoroughly in Section 5.6.

are substantial, the issues raised from the modeling of human players define a far more complex and important problem for the understanding of player experience.

Sometimes the terms player modeling and **opponent modeling** [214, 592, 48] are used interchangeably when a human player is modeled. However, opponent modeling is a more narrow concept referring to predicting behavior of an adversarial player when playing to win in an imperfect information game like Poker [48] or *StarCraft* (Blizzard Entertainment, 1988) [504]. Some aspects of modeling NPCs or simulated playthroughs for winning in a game are discussed in Chapter 3.

We also make a distinction between player modeling [116, 281] and **player profiling** [782]. The former refers to modeling complex **dynamic** phenomena during gameplay interaction, whereas the latter refers to the categorization of players based on **static** information that does not alter during gameplay. Information of static nature includes personality, cultural background, gender and age. We put an emphasis on the former, but will not ignore the latter, as the availability of a good player profile may contribute to the construction of reliable player models.

In summary, player modeling—as we define it in this book—is the study of computational means for the modeling of a player’s experience or behavior which is based on theoretical frameworks about player experience and/or data derived from the interaction of the player with a game [782, 764]. Player models are built on dynamic information obtained during game-player interaction, but they could also rely on static player profiling information. Unlike studies focusing on taxonomies of **behavioral** player modeling—e.g., via a number of dimensions [636] or direct/indirect measurements [623]—we view player modeling in a holistic manner including cognitive, affective, personality and demographic aspects of the player. Moreover, we exclude approaches that are not directly based on human-generated data or not based on empirically-evaluated theories of player experience, human cognition, affect or behavior. The chapter does not intend to provide an exhaustive review of player modeling studies under the above definition, but rather an introduction and a high-level taxonomy that explores the possibilities with respect to the modeling approach, the model’s input and the model’s output.

## 5.2 Why Model Players?

The *primary goal* of player modeling is to *understand* how the interaction with a game is experienced by individual players. Thus, while games can be utilized as an arena for eliciting, evaluating, expressing and even synthesizing experience, we argue that the main aim of the study of players in games is the understanding of players’ cognitive, affective and behavioral patterns. Indeed, by the very nature of games, one cannot dissociate games from player experience.

There are two core reasons that drive the use of AI for modeling game players and their play, thereby serving the primary goal of player modeling as stated above. The *first* is for understanding something about their players’ **experience** during play. Models of player experience are often built using machine learning methods, typically supervised learning methods like support vector machines or neural networks. The training data here consists of some aspect of the game or player-game interaction, and the targets are labels derived from some assessment of player experience, gathered for example from physiological measurements or questionnaires [781]. Once predictors of player experience are derived they can be taken into account for designing the in-game experience. That can be achieved by adjusting the behavior of non-player characters (see Chapter 3) or by adjusting the game environment (see Chapter 4).

The *second* reason why one would want to use AI to model players is for understanding players’ **behavior** in the game. This area of player modeling is concerned with structuring observed player behavior even when no measures of experience are available—for instance, by identifying player types or predicting player behavior via game and player **analytics** [178, 186]. A popular distinction in data derived from games [186] is the one between **player metrics** and **game metrics**. The latter is a superset of the former as it also includes metrics about the game software (system

metrics) and the game development process as a whole (process metrics). System metrics and process metrics are important aspects of modern game development that influence decision making with respect to procedures, business models, and marketing. In this book, however, we focus on player metrics. The interested reader may refer to [186] for alternative uses of metrics in games and the application of analytics to game development and research—i.e., **game analytics**.

Once aspects of player behavior are identified a number of actions can be taken to improve the game such as the personalization of content, the adjustment of NPCs or, ultimately, the redesign of (parts of) the game. Derived knowledge about the in-game behavior of the player can lead to improved game testing and game design procedures, and better monetization and marketing strategies [186]. Within behavior modeling we identify four main player modeling subtasks that are particularly relevant for game AI: **imitation** and **prediction**—achieved via supervised learning or reinforcement learning—and **clustering** and **association mining**—achieved via unsupervised learning. The two main purposes of player **imitation** is the development of non-player characters with believable, human-like behavioral characteristics, and the understanding of human play per se through creating generative models of it. The **prediction** of aspects of player behavior, instead, may provide answers to questions such as “when will this player stop playing?” or “how often will that player get stuck in that area of the level?” or “which item type will this player pick in the next room?”. The aim of **clustering** is the classification of player behaviors within a number of clusters depending of their behavioral attributes. Clustering is important for both the personalization of the game and the understanding of playing behavior in association with the game design [178]. Finally, **association mining** is useful in instances where frequent patterns or sequences of actions (or in-game events) are important for determining how a player behaves in a game.

While player behavior and player experience are interwoven notions there is a subtle difference between them. Player behavior points to **what a player does** in a game whereas player experience refers to **how a player feels** during play. The feeling of one’s gameplay experience is clearly associated with what one does in the game; player experience, however, is primarily concerned with affective and cognitive aspects of play as opposed to mere reactions of gameplay which refer to player behavior.

Given the above aims, core tasks and sub-tasks of player modeling in the next section we discuss the various available options for constructing a player model.

### 5.3 A High-Level Taxonomy of Approaches

Irrespective of the application domain, computational models are characterized by three core components: the input the model will consider, the computational model per se, and the output of the model (see Fig. 5.1). The model itself is a mapping between the input and the output. The mapping is either hand-crafted or derived from data, or a mix of the two. In this section we will first go through the most

common approaches for constructing a computational model of players, then we will go through a taxonomy of possible inputs for a player model (Section 5.4) and finally we will examine aspects of player experience and behavior that a player model can represent as its output (Section 5.5).

A high-level classification of the available approaches for player modeling can be made between **model-based** (or top-down) and **model-free** (or bottom-up) approaches [782, 783]. The above definitions are inspired by the analogous classification in RL by which a world model is available (i.e., model-based) or not (i.e., model-free). Given the two ends of this continuum **hybrid** approaches between them can naturally exist. The gradient red color of the player model box in Fig. 5.1 illustrates the continuum between top-down and bottom-up approaches. The remainder of this section presents the key elements of and core differences among the various approaches for modeling of players.

### **5.3.1 Model-Based (Top-Down) Approaches**

In a **model-based** or top-down [782] approach a player model is built on a theoretical framework. As such, researchers follow the modus operandi of the humanities and social sciences, which hypothesize models to explain phenomena. Such hypotheses are usually followed by an empirical phase in which it is experimentally determined to what extent the hypothesized models fit observations; however, such a practice is not the norm within player experience research. While user experience has been studied extensively across several disciplines, in this book we identify three main disciplines we can borrow theoretical frameworks from and build models of player experience: **psychology and affective sciences**, **neuroscience**, and finally, **game studies and game research**.

#### **5.3.1.1 Psychology and Affective Sciences**

Top-down approaches to player modeling may refer to models derived from popular theories about emotion [364] such as the cognitive appraisal theory [212, 601]. Further, the player model may rely on well established affect representations such as the emotional dimensions of arousal and valence [200] that define the circumplex model of affect of Russell [539] (see Fig. 5.2(a)). Valence refers to how pleasurable (positive) or unpleasurable (negative) the emotion is whereas arousal refers to how intense (active) or lethargic (inactive) that emotion is. Following a theoretical model, emotional manifestations of players are often mapped directly to specific player states. For instance, by viewing player experience as a psychophysiological phenomenon [779] a player's increased heart rate may correspond to high arousal and, in turn, to high levels of excitement or frustration.

Beyond established theories of emotion, model-based approaches can also be inspired by a general cognitive-behavioral theoretical framework such as the *theory*

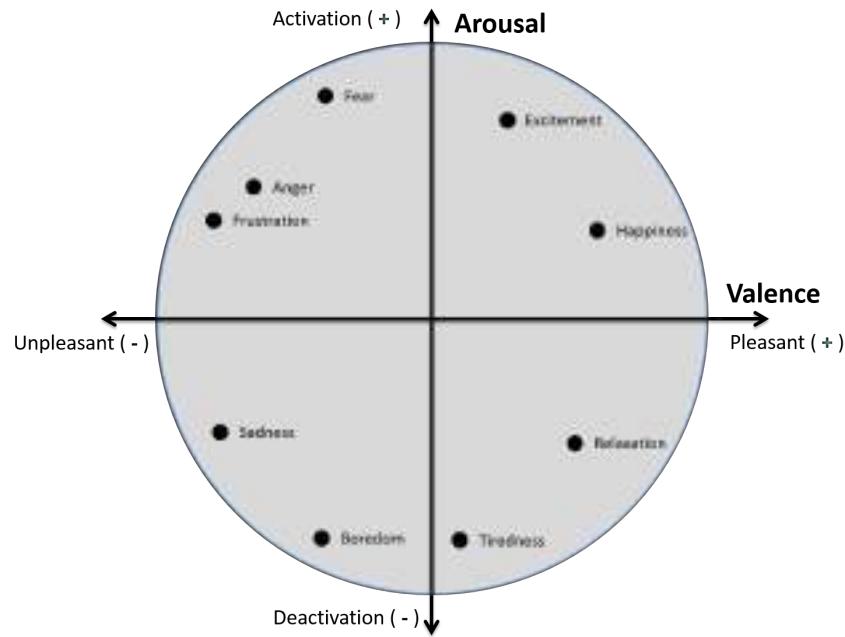
*of mind* [540] for modeling aspects of social interactions in games. Popular example frameworks for deriving user models in games include the usability theory [489, 290], the belief-desire-intention (BDI) model [66, 224], the cognitive model by Ortony, Clore, and Collins [512] and Skinner's behavioristic approach [633] with its links to reward systems in games. Further we can draw inspiration from social sciences and linguistics in order to model lexical aspects of gameplay interaction (e.g., chatting). Natural language processing, opinion mining and sentiment analysis are normally relying on theoretical models that build on affective and sociological aspects of textual communication [517, 514].

Of particular importance is the concept of **flow** by Csikszentmihalyi [151, 149, 150] which has been a popular psychological construct for modeling player experience in a top-down fashion. When in a state of flow (or else, state of "happiness") during an activity we tend to concentrate on the present moment, we lose our ability of reflective self-consciousness, we feel a sense of personal control over the situation or activity, our perception of time is altered, and we experience the activity as intrinsically rewarding. Analogously the optimal experience during play has been associated with a fine balance between boredom and anxiety, also known as the **flow channel** (see Fig. 5.2(b)). Given its direct relevance to player experience, flow has been adapted and incorporated for use in game design and for the understanding of player experience [678, 675, 473].

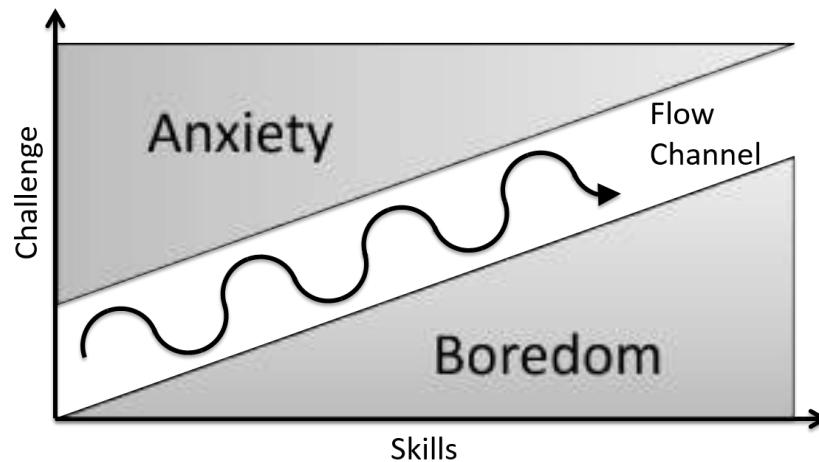
### 5.3.1.2 Neuroscience

A number of studies have relied on the working hypothesis of an underlying mapping between the brain, its neural activity and player experience. However, this relationship is not well explored and the presumptive mapping is largely unknown. For example, interest has been associated with activity in the visual cortex and the release of endorphin whereas the sense of achievement has been linked to dopamine levels [35]. According to [35], neuroscientific evidence suggests that the reward systems of games are directly associated with the dopamine-based reward structures in the brain and that dopamine is released during gameplay [346]. Further, pleasure has been associated with areas in the brain responsible for decision making, thereby revealing the direct links between gameplay experience and decision making [575]. Pleasure has also been associated with uncertain outcomes or uncertain rewards [625] as well as with interest and curiosity [43], which are all key elements of successful game design. Stress is also tightly coupled with player experience given its clear association with anxiety and fear; stress can be both monitored via physiology and regulated via game design. The testosterone levels of players have also been measured in association to digital game activities [444] and findings reveal particular patters of competition in games as testosterone factors. Finally, it appears that trust between players in a social gaming setup could be measured indirectly via oxytocin levels [350].

The degree to which current findings from neuroscience are applicable to player experience research is largely unknown since access to neural activity and brain hor-



(a) Russell's two-dimensional circumplex model of affect. The figure contains a small number of representative affective states (black circles).



(b) An illustration of the *flow channel*.

**Fig. 5.2** Two popular frameworks used for modeling users and their experience in games: (a) the arousal-valence circumplex model of affect and (b) the flow channel concept.

mone levels remains a rather intrusive process at the time of writing. Manifestations of brain activity such as the brain's electrical waves—measured through electroencephalography on our scalp—or more indirect manifestations such as stress and anxiety—measured through skin conductance—can give us access to approximates of brain activity. These approximates can be used for modeling the experience of play as discussed later in this chapter.

### 5.3.1.3 Game Studies and Game Research

Theoretical models of user experience in games are often driven by work in game studies and game research. Examples of models that have been used extensively in the literature include Malone's core design dimensions that collectively contribute to 'fun' games [419] defined as **challenge**, **curiosity** and **fantasy**. In particular, challenge refers to the uncertainty of achieving a goal due to e.g., variable difficulty level, multiple level goals, hidden information, and randomness. Curiosity refers to the player's feeling of uncertainty with respect to what will happen next. Finally, fantasy is the ability of the game to show (or evoke) situations or contexts that are not actually present. These three dimensions have been quantified, operationalized and successfully evaluated in prey-predator games [766], physical games [769, 775], preschooler games [320] and racing games [703].

Bartle's [33] classification of player types within games as a form of *general* player profiles can be used indirectly for modeling players. Bartle identifies four archetypes of players he names **killers** (i.e., players that focus on winning and are engaged by ranks and leaderboards), **achievers** (i.e., players that focus on achieving goals quickly and are engaged by achievements), **socializers** (i.e., players that focus on social aspects of games such as developing a network of friends) and **explorers** (i.e., players who focus on the exploration of the unknown). Various other methodologies have also been followed to derive *specific* player experience archetypes for particular classes of games [34, 787].

Other popular and interconnected views of player experience from a *game design* perspective include the theory of 'fun' by Koster [351], the notion of the 'magic circle' in games [587] and the four "fun" factor model of Lazzaro [365]. Indicatively, Koster's theory relates the concept of **fun** with learning in games: the more you learn the more you tend to play a game. According to his theory you stop playing a game that is way too easy (no learning of new skills) or way too hard (no learning either). Lazzaro's four fun factors are named **hard fun** (e.g., playing to win and see how good I am at it), **easy fun** (e.g., playing to explore new worlds and game spaces), **serious fun** (e.g., playing to feel better about myself or get better at something that matters to me) and **people fun** (e.g., playing as an excuse to invite friends over, or having fun merely by watching them play). Within *game studies*, the theoretical model of incorporation [94] is a notable multifaceted approach for capturing player immersion. The model is composed of six types of player involvement: affective, kinaesthetic, spatial, ludic, and narrative.

With a careful analysis of the models proposed and their subcomponents one could coherently argue that there is one underlying theoretical model of player experience after all. While it is not the intention of this book to thoroughly discuss the interconnections between the aforementioned models it is worth pointing out a number of indicative examples of our envisaged overarching player experience model. An explorer (Bartle), for instance, can be associated with the easy fun factor of Lazzaro and the curiosity dimension of Malone. Further, the achiever archetype (Bartle) can be linked to the serious fun factor (Lazzaro). Accordingly, a killer archetype (Bartle) maps to the hard fun factor (Lazzaro), the challenge dimension of Malone's model, and a number of flow aspects. Finally, a socializer player profile (Bartle) could be associated to people fun (Lazzaro) and, in turn, to the shared involvement facet of Calleja [94].

Even though the literature on theoretical models of experience is rather rich, one needs to be cautious with the application of such theories to games (and game players) as the majority of the models have not been derived from or tested on interactive media such as games. Calleja [94], for instance, reflects on the inappropriateness of the concepts of 'fun' and 'magic circle' (among others) for games. At this point it is worth noting that while ad-hoc designed models can be an extremely powerful and expressive they need to be cross-validated empirically to be of practical use for computational player modeling; however, such practices are not as common within the broader area of game studies and game design.

### **5.3.2 Model-Free (Bottom-Up) Approaches**

**Model-free** approaches refer to the data-driven construction of an unknown mapping (model) between a player **input** and a **player state**. Any manifestation of player affect or behavioral pattern could define the input of the model (see more in Section 5.4 below). A player state, on the other hand, is any representation of the player's experience or current emotional, cognitive, or behavioral state; this is essentially the output of the computational model (see more in Section 5.5). Evidently, model-free approaches follow the modus operandi of the exact sciences, in which observations are collected and analyzed to generate models without a strong initial assumption on what the model looks like or even what it captures. Player data and labels of player states are collected and used to derive the model.

Classification, regression and preference learning techniques adopted from machine learning—see Chapter 2—or statistical approaches are commonly used for the construction of the mapping between the input and the output. Examples include studies in which player actions, goals and intentions are modeled and predicted for the purpose of believable gameplay, adaptive gameplay, interactive storytelling or even the improvement of a game's monetization strategy [511, 800, 414, 693, 592]. In contrast to supervised learning, reinforcement learning can be applied when a reward function, instead, can characterize aspects of playing behavior or experience. Unsupervised learning is applicable when target outputs are not available for pre-

dictive purposes but, alternatively, data is used for the analysis of playing behavior (see Fig. 5.1).

We meet bottom-up player modeling attempts since the early years of the game AI field in first-person shooters [695, 696], racing games [703] and variants of *Pac-Man* (Namco, 1980) [776]. Recently, the availability of large sets of game and player data has opened up the horizons of behavioral data mining in games—i.e., **game data mining** [178]. Studies that attempt to identify different behavioral, playing and action patterns within a game are well summarized in [186] and include [36, 176, 687, 690, 750], among many others.

### 5.3.3 Hybrids

The space between a completely model-based and a completely model-free approach can be viewed as a continuum along which any player modeling approach might be placed. While a completely model-based approach relies solely on a theoretical framework that maps a player’s responses to game stimuli, a completely model-free approach assumes there is an unknown function between modalities of user input and player states that a machine learner (or a statistical model) may discover, but does not assume anything about the structure of this function. Relative to these extremes, the vast majority of studies in player modeling may be viewed as **hybrids** that synergistically combine elements of the two approaches. The continuum between top-down and bottom-up player modeling approaches is illustrated with a gradient color in Fig. 5.1.

## 5.4 What Is the Model’s Input Like?

By now we have covered the various approaches available for modeling players and we will, in this section, focus on what the input of such a model might be like. The model’s input can be of three main types: (1) anything that a player is doing in a game environment gathered from **gameplay** data—i.e., behavioral data of any type such as user interface selections, preferences, or in-game actions; (2) **objective** data collected as responses to game stimuli such as physiology, speech and body movements; and (3) the **game context** which comprises of any player-agent interactions but also any type of game content viewed, played, and/or created. The three input types are detailed in the remainder of this section. At the end of the section we also discuss *static* profile information on the player (such as personality) as well as web data beyond games that could feed and enhance the capacity of a player model.

### 5.4.1 Gameplay

Given that games may affect the player's cognitive processing patterns and cognitive focus we assume that a player's actions and preferences are linked directly to her experience. Consequently, one may infer the player's current experience by analyzing patterns of her interaction with the game, and by associating her experience with game context variables [132, 239]. Any element derived from the direct interaction between the player and the game can be classified as **gameplay** input. These interpretable measures of gameplay have also been defined as **player metrics** [186]. Player metrics include detailed attributes of the player's behavior derived from responses to game elements such as NPCs, game levels, user menus, or embodied conversational agents. Popular examples of data attributes include detailed spatial locations of players viewed as **heatmaps** [177], statistics on the use of in-game menus, as well as descriptive statistics about gameplay, and communication with other players. Figure 5.3 shows examples of heatmaps in the *MiniDungeons*<sup>1</sup> puzzle game. Both general measures (such as performance and time spent on a task) and game-specific measures (such as the weapons selected in a shooter game [250]) are relevant and appropriate player metrics.

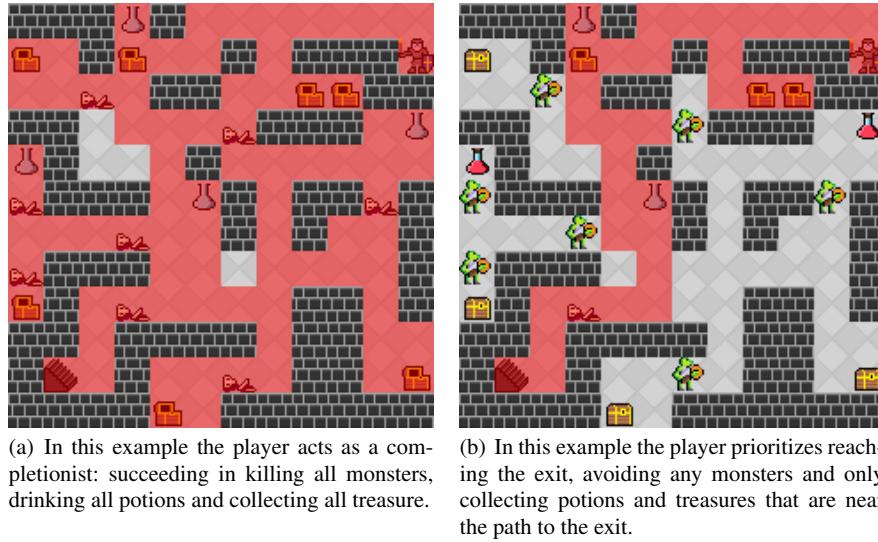
A major limitation with the gameplay input is that the actual player experience is only **indirectly** observed. For instance, a player who has little interaction with a game might be thoughtful and captivated, or just bored and busy doing something else. Gameplay metrics can only be used to approach the likelihood of the presence of certain player experiences. Such statistics may hold for player populations, but may provide little information for individual players. Therefore, when one attempts to use pure player metrics to make estimates of player experiences and make the game respond in an appropriate manner to these perceived experiences, it is advisable to keep track of the feedback of the player to the game responses, and adapt when the feedback indicates that the player experience was gauged incorrectly.

### 5.4.2 Objective

Computer game players are presented with a wide palette of affective stimuli during game play. Those stimuli vary from simple auditory and visual events (such as sound effects and textures) to complex narrative structures, virtual cinematographic views of the game world and emotively expressive game agents. Player emotional responses may, in turn, cause changes in the player's physiology, reflect on the player's facial expression, posture and speech, and alter the player's attention and focus level. Monitoring such bodily alterations may assist in recognizing and constructing the player's model. As such, the **objective** approach to player modeling incorporates access to multiple modalities of player input.

---

<sup>1</sup> <http://minidungeons.com/>



**Fig. 5.3** Two example heatmaps (human playtraces) in the *MiniDungeons* game. *MiniDungeons* is a simple turn-based rogue-like puzzle game, implemented as a benchmark problem for modeling the decision-making styles of human players [267].

The relationship between psychology and its physiological manifestations has been studied extensively ([17, 95, 779, 558] among many others). What is widely evidenced is that the sympathetic and the parasympathetic components of the autonomic nervous system are involuntary affected by affective stimuli. In general, arousal-intense events cause dynamic changes in both nervous systems: an increase and a decrease of activity, respectively, in the sympathetic and the parasympathetic nervous system. Alternatively, activity at the parasympathetic nervous system is high during relaxing or resting states. As mentioned above, such nervous system activities cause alterations in one's facial expression, head pose, electrodermal activity, heart rate variability, blood pressure, pupil dilation [91, 624] and so on.

Recent years have seen a significant volume of studies that explore the interplay between **physiology** and gameplay by investigating the impact of different gameplay stimuli on dissimilar physiological signals ([697, 473, 421, 420, 556, 721, 175, 451] among others). Such signals are usually obtained through electrocardiography (ECG) [780], photoplethysmography [780, 721], galvanic skin response (GSR) [421, 271, 270, 272], respiration [721], electroencephalography (EEG) [493] and electromyography (EMG).

In addition to physiology one may track the player's **bodily expressions** (motion tracking) at different levels of detail and infer the real-time affective responses from the gameplay stimuli. The core assumption of such input modalities is that particular bodily expressions are linked to expressed emotions and cognitive processes. Objective input modalities, beyond physiology, that have been explored ex-

tensively include facial expressions [321, 19, 236, 88, 794], muscle activation (typically face) [133, 164], body movement and posture [23, 731, 321, 172, 47], speech [741, 319, 308, 306, 30], text [517, 137, 391], haptics [509], gestures [283], brain waves [559, 13], and eye movement [23, 469].

While objective measurements can be a very informative way of assessing the player's state during the game a major limitation with most of them is that they can be invasive, thus affecting the player's experience with the game. In fact, some types of objective measures appear to be **implausible** within commercial-standard game development. **Pupillometry** and **gaze tracking**, for instance, are very sensitive to distance from screen, and variations in light and screen luminance, which collectively make them rather impractical for use in a game application. The recent rebirth of virtual reality (VR), however, gives eye gaze sensing technologies entirely new opportunities and use within games [628]; a notable example of a VR headset that features eye-tracking is FOVE.<sup>2</sup> Other **visual cues** obtained through a camera (facial expressions, body posture and eye movement) require a well-lit environment which is often not present in home settings (e.g., when playing video-games) and they can be seen by some players as privacy hazards (as the user is continuously recorded). Even though highly unobtrusive, the majority of the vision-based affect-detection systems currently available have additional limitations when asked to operate in real-time [794]. We argue that an exception to this rule is body posture, which can both be effectively detected nowadays and provide us with meaningful estimates of player experience [343]. Aside from the potential they might have, however, the appropriateness of camera-based input modalities for games is questionable since experienced players tend to stay still while playing games [22].

As a response to the limitations of camera-based measurements, **speech** and **text** (e.g., chat) offer two highly accessible, real-time efficient and unobtrusive modalities with great potential for gaming applications; however, they are only applicable to games where speech (or text) forms a control modality (as e.g., in conversational games for children [320, 789]), collaborative games that naturally rely on speech or text for communication across players (e.g., in collaborative first-person shooters), or games that rely on natural language processing such as text-based adventure games or interactive fiction (see discussion of Chapter 4).

Within players' **physiology**, existing hardware for EEG, respiration and EMG require the placement of body parts such as the head, the chest or parts of the face on the sensors, making those physiological signals rather impractical and highly intrusive for most games. On the contrary, recent sensor technology advancements for the measurement of electrodermal activity (skin conductivity), photoplethysmography (blood volume pulse), heart rate variability and skin temperature have made those physiological signals more attractive for the study of affect in games. Real-time recordings of these can nowadays be obtained via comfortable wristbands and stored in a personal computer or a mobile device via a wireless connection [779].

At the moment of writing there are a few examples of **commercial games** that utilize physiological input from players. One particularly interesting example is

---

<sup>2</sup> <https://www.getfove.com/>



**Fig. 5.4** A screenshot from *Nevermind* (Flying Mollusk, 2015). The game supports several off-the-shelf sensors that allow the audiovisual content of the game to adapt to the stress levels of the player. Image obtained from Erin Reynolds with permission.

*Nevermind* (Flying Mollusk, 2015), a biofeedback-enhanced adventure horror game that adapts to the player's stress levels by increasing the level of challenge it provides: the higher the stress the more the challenge for the player (see Fig. 5.4). A number of sensors which detect heart activity are available for affective interaction with *Nevermind*. *The Journey of Wild Divine* (Wild Divine, 2001) is another biofeedback-based game designed to teach relaxation exercises via the player's blood volume pulse and skin conductance. It is also worth noting that AAA game developers such as Valve have experimented with the player's physiological input for the personalization of games such as *Left 4 Dead* (Valve, 2008) [14].

### 5.4.3 Game Context

In addition to gameplay and objective input, the game's context is a necessary input for player modeling. **Game context** refers to the momentaneous state of the game during play and excludes any gameplay elements; those are already discussed in the gameplay input section. Clearly, our gameplay affects some aspects of the game context and vice versa but the two can be viewed as separate entities. Viewing this relationship from an analytics lens, the game context can be seen as a form of game metrics, opposed to gameplay which is a form of player metrics.

The importance of the game context for modeling players is obvious. In fact, we could argue that the context of the game during the interaction is a necessary input for detecting reliably any cognitive and affective responses of players. It could also

be argued that the game context is necessary as a guide during the annotation of the player experience; but more of that we will discuss in Section 5.5. The same way that we require the current social and cultural context to better detect the underlying emotional state of a particular facial expression of our discussant any player reactions cannot be dissociated from the stimulus (or the game context) that elicited them. Naturally, player states are always linked to game context. As a result, player models that do not take context into account run a risk of inferring erroneous states for the player. For example, an increase in galvanic skin response can be linked to different high-arousal affective states such as frustration and excitement. It is very hard to tell however, what the heightened galvanic skin response “means” without knowing what is happening in the game at the moment. In another example, a particular facial expression of the player, recorded though a camera, could be associated with either an achievement in the game or a challenging moment, and needs to be triangulated with the current game state to be understood. Evidently, such dualities of the underlying player state may be detrimental for the design of the player model.

While a few studies have investigated physiological reactions of players in isolation, good practice in player modeling commands that any reactions of the players is triangulated with information about the current game state. For instance, the model needs to know if the GSR increases because the player died or completed the level. The game context—naturally combined (or **fused**) with other input modalities from the player—has been used extensively in the literature for the prediction of different affective and cognitive states relevant to playing experience [451, 434, 521, 617, 572, 133, 254, 558, 452, 433].

#### 5.4.4 Player Profile

A **player profile** includes all the information about the player which is *static* and it is not directly (nor necessarily) linked to gameplay. This may include information on player personality (such as expressed by the Five Factor Model of personality [140, 449]), culture dependent factors, and general demographics such as gender and age. A player’s profile may be used as input to the player model to complement the captured in-game behavior with general attributes about the player. Such information may lead to more precise predictive models about players.

While gender, age [787, 686], nationality [46] and player expertise level [96] have already proven important factors for profiling players the role of personality remains somewhat contentious. On the one hand, the findings of van Lankveld et al. [736, 737], for instance, reveal that gameplay behavior does not necessarily correspond to a player’s behavior beyond the game. On the other hand, Yee et al. have identified strong correlations between player choices in *World of Warcraft* (Blizzard Entertainment, 2004) and the personalities of its players [788]. Strong correlations have also been found between the playing style and personality in the first-person shooter *Battlefield 3* (Electronic Arts, 2011) [687]. In general, we need to acknowledge that there is no guaranteed one-to-one mapping between a player’s in-game

behavior and personality, and that a player’s personality profile does not necessarily indicate what the player would prefer or like in a game [782].

#### 5.4.5 Linked Data

Somewhere between the highly dynamic in-game behavior and the static profile information about the player we may also consider **linked data** retrieved from web services that are not associated with gameplay per se. This data, for instance, may include our social media posts, emoticons, emojis [199], tags used, places visited, game reviews written, or any relevant semantic information extracted from diverse Web content. The benefit of adding such information to player models is many-fold but it has so far seen limited use in games [32]. In contrast to current player modeling approaches the use of massive amounts and dissimilar types of content across linked online sources would enable the design of player models which are based on user information stored across various online datasets, thereby realizing semantically-enriched game experiences. For example, both scores and sentiment-analyzed textual reviews [517, 514] from game review sites such as Metacritic<sup>3</sup> or GameRankings<sup>4</sup> can be used as input to a model. This model can then be used to create game content which is expected to appeal to the specific parts of the community, based, for instance, on demographics, skill or interests collected from the user’s in-game achievements or favored games [585].

### 5.5 What Is the Model’s Output Like?

The model’s **output**, i.e., that which we wish to model, is usually a representation of the player’s state. In this section we explore three options for the output of the model that serve different purposes in player modeling. If we wish to model the **experience** of the player the output is provided predominately through manual annotation. If instead we wish to model aspects of player **behavior** the output is predominately based on in-game actions (see Fig. 5.1). Finally, it may very well be that the model has **no output**. Section 5.5.1 and Section 5.5.2 discuss the particularities of the output, respectively, for the purpose of behavioral modeling and experience modeling whereas Section 5.5.3 explores the condition where the model has no outputs.

---

<sup>3</sup> <http://www.metacritic.com>

<sup>4</sup> <http://www.gamerankings.com/>

### 5.5.1 Modeling Behavior

The task of modeling player behavior refers to the prediction or imitation of a particular behavioral state or a set of states. Note that if no target outputs are available then we are faced with either an unsupervised learning problem or a reinforcement learning problem which we discuss in Section 5.5.3. The output we must learn to predict (or imitate) in a supervised learning manner can be of two major types of **game-play** data: either **micro-actions** or **macro-actions** (see Fig. 5.1). The first machine learning problem considers the moment-to-moment game state and player action space that are available at a frequency of frame rates. For example, we can learn to imitate the moves of a player on a frame-to-frame basis by comparing the play traces of an AI agent and a human as e.g., done for *Super Mario Bros* (Nintendo, 1985) [511, 469]. When macro-actions are considered instead, the target output is normally an aggregated feature of player behavior over time, or a behavioral pattern. Examples of such outputs include game completion times, win rates, churn, trajectories, and game balance.

### 5.5.2 Modeling Experience

To model the experience of the player one needs to have access to labels of that experience. Those labels ideally need to be as close to the **ground truth** of experience as possible. The ground truth (or gold standard) in affective sciences refers to a hypothesized and unknown label, value, or function, that best characterizes and represents an affective construct or an experience. Labels are normally provided through manual annotation which is a rather laborious process. Manual annotation is however necessary given that we require some estimate of the ground truth for subjective notions such as the emotional states of the player. The **accuracy** of that estimation is regularly questioned as there are numerous factors contributing to a deviation between a label and the actual underlying player experience.

Manually annotating players and their gameplay is a challenge in its own right with respect to both the human annotators involved and the annotation protocol chosen [455, 777]. On one hand, the annotators need to be skilled enough to be able to approximate the actual experience well. On the other hand, there are still many open questions left for us to address when it comes to the annotation tools and protocols used. Such questions include: Who will do the labeling: the person experiencing the gameplay or others? Will the labeling of player experience involve states (discrete representation) or instead involve the use of intensity or experience dimensions (continuous representation)? When it comes to time, should it be done in real-time or offline, in discrete time periods or continuously? Should the annotators be asked to rate the affect in an absolute fashion or, instead, rank it in a relative fashion? Answers to the above questions yield different data annotation protocols and, inevitably, varying degrees of data quality, validity and reliability. In the following

sections we attempt to address a number of such critical questions that are usually raised in subjective annotations of player states.

### 5.5.2.1 Free Response or Forced Response?

Subjective player state annotations can be based either on a player’s **free response**—retrieved via e.g., a think-aloud protocol [555]—or on **forced responses** retrieved through questionnaires or annotation tools. Free response naturally contains richer information about the player’s state, but it is often unstructured, even chaotic, and thus hard to analyze appropriately. On the other hand, forcing players to self-report their experiences using directed questions or tasks constrains them to specific questionnaire items which could vary from simple tick boxes to multiple choice items. Both the questions and the answers we provide to annotators may vary from single words to sentences. Questionnaires can contain elements of player experience (e.g., the Game Experience Questionnaire [286]), demographic data and/or personality traits (e.g., a validated psychological profiling questionnaire such as the NEO-PI-R [140]). In the remainder of this section we will focus on forced responses as these are easier to analyze and are far more appropriate for data analysis and player modeling (as defined in this book).

### 5.5.2.2 Who Annotates?

Given the subjective nature of player experience the first natural question that comes in mind is *who annotates players?* In other words, who has the right authority and the best capacity to provide us with reliable tags of player experience? We distinguish two main categories: annotations can either be self-reports or reports expressed indirectly by experts or external observers [783].

In the first category the player states are provided by the players themselves and we call that **first-person** annotation. For example, a player is asked to rate the level of engagement while watching her playthrough video. First-person is clearly the most direct way to annotate a player state and build a model based on the solicited annotations. We can only assume there is disparity between the true (inner) experience of each player and the experience as felt by herself or perceived by others. Based on this assumption the player’s annotations should normally be closer to her inner experience (**ground truth**) compared to third-person annotation. First-person annotation, however, may suffer from self-deception and memory limitations [778]. These limitations have been attributed mainly to the discrepancies between “the experiencing self” and “the remembering self” of a person [318] which is also known as the **memory-experience gap** [462].

Expert annotators—as a response to the above limitations—may instead be able to surpass the perception of experience and reach out to the inner experience of the player. In this second annotation category, named **third-person** annotation, an expert—such as a game designer—or an external observer provides the player state

in a more objective manner, thereby reducing the subjective biases of first-person perceptions. For instance, a user experience analyst may provide particular player state tags while observing a first-person shooter deathmatch game. The benefit of third-person annotation is that multiple annotators can be used for a particular gameplay experience. In fact, the availability of several such subjective perceptions of experience may allow us to approximate the ground truth better as the agreement between many annotators enhances the validity of our data directly. A potential disagreement, on the other hand, might suggest that the gameplay experience we examine is non-trivial or may indicate that some of our annotators are untrained or inexperienced.

### 5.5.2.3 How Is Player Experience Represented?

Another key question is how player experience is best represented: as a number of different states (**discrete**) or, alternatively, as a set of dimensions (**continuous**)? On one hand, discrete labeling is practical as a means of representing player experience since the labels can easily form individual items (e.g., “excited”, “annoyed” etc.) in an experience questionnaire, making it easy to ask the annotator/player to pick one (e.g., in [621]). Continuous labeling, on the other hand, appears to be advantageous for two key reasons. First, experiential states such as *immersion* are hard to capture with words or linguistic expressions that have fuzzy boundaries. Second, states do not allow for variations in *experience intensity* over time since they are binary: either the state is present or not. For example, the complex notions of *fun*, or even *engagement*, cannot be easily captured by their corresponding linguistic representation in a questionnaire or define well a particular *state* of a playing experience. Instead it seems natural to represent them as a continuum of experience intensity that may vary over time. For these reasons we often observe low agreement among the annotators [143] when we represent playing experience via discrete states.

As discussed earlier, the dominant approach in continuous annotation is the use of Russell’s two-dimensional (arousal-valence) circumplex model of affect [581] (see Fig. 5.2(a)). Figure 5.5 illustrates two different annotation tools (FeelTrace and *AffectRank*) that are based on the arousal-valence circumplex model of affect. Figure 5.6 depicts the *RankTrace* continuous annotation tool which can be used for the annotation of a single dimension of affect (i.e., tension in this example). All three tools are accessible and of direct use for annotating playing experience.

### 5.5.2.4 How Often to Annotate?

Annotation can happen either within particular time intervals or continuously. **Time-continuous** annotation has been popularized due to the existence of freely available tools such as FeelTrace [144] (see Fig. 5.5(c)) and GTrace [145], which allows for continuous annotation of content (mostly videos and speech) across the dimensions of arousal and valence. In addition to FeelTrace there are annotation tools like the

continuous measurement system [454] and EmuJoy [474], where the latter is designed for the annotation of music content. User interfaces such as wheels and knobs linked to the above annotation tools show further promise for the continuous annotation of experience in games [125, 397, 97] (see Fig. 5.6). The continuous annotation process, however, appears to require a higher amount of cognitive load compared to a time-discrete annotation protocol. Higher cognitive loads often result in lower levels of agreement between different annotators and yield unreliable data for modeling player experience [166, 418].

As a response to the above limitations, **time-discrete** annotation provides data at particular intervals when the annotator feels there is a *change* in the player’s state. And changes are best indicated *relatively* rather than *absolutely*. *AffectRank*, for instance (see Fig. 5.5(b)), is a discrete, rank-based annotation tool that can be used for the annotation of any type of content including images, video, text or speech and it provides annotations that are significantly more reliable (with respect to inter-rater agreement) than the annotations obtained from continuous annotation tools such as *FeelTrace* [777]. The rank-based design of *AffectRank* is motivated by observations of recent studies in third-person video annotation indicating that “...humans are better at rating emotions in *relative* rather than absolute terms.” [455, 777]. Further, *AffectRank* is grounded in numerous findings showcasing the supremacy of ranks over ratings for obtaining annotations of lower inconsistency and order effects [777, 773, 778, 436, 455, 761].

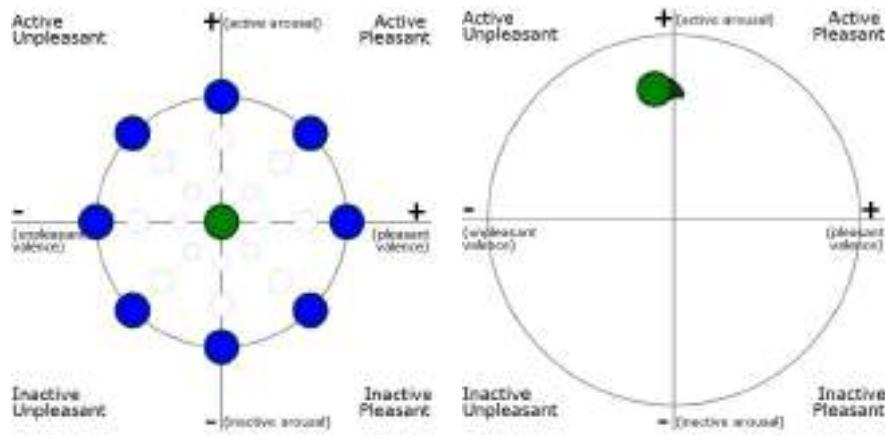
A recent tool that builds on the relative-based annotation of *AffectRank* and allows for the annotation of affect in a continuous yet unbounded fashion is *RankTrace* (see Fig. 5.6). The core idea behind *RankTrace* is introduced in [125]: the tool asks participants to watch the recorded playthrough of a play session and annotate in real-time the perceived intensity of a single emotional dimension. The annotation process in *RankTrace* is controlled through a “wheel-like” hardware, allowing participants to meticulously increase or decrease emotional intensity by turning the wheel, similarly to how volume is controlled in a stereo system. Further, the general interfacing design of *RankTrace* builds on the one-dimensional GTrace annotation tool [145]. Unlike other continuous annotation tools, however, annotation in *RankTrace* is *unbounded*: participants can continuously increase or decrease the intensity as desired without constraining themselves to an absolute scale. This design decision is built on the *anchor* [607] and *adaptation level* [258] psychology theories by which affect is a temporal notion based on earlier experience that is best expressed in relative terms [765, 777, 397]. The use of *RankTrace* has revealed the benefits of relative and unbounded annotation for modeling affect more reliably [397] and has also showed promise for the construction of general models of player emotion across games [97].

### 5.5.2.5 When to Annotate?

When is it best to annotate experience: *before*, *during* or *after* play (see Fig. 5.1)? In a **pre**-experience questionnaire we usually ask annotators to set the baseline of



(a) Annotating facial expressions of players during play. The annotation is context-dependent as the video includes the gameplay of the player (see top left corner).

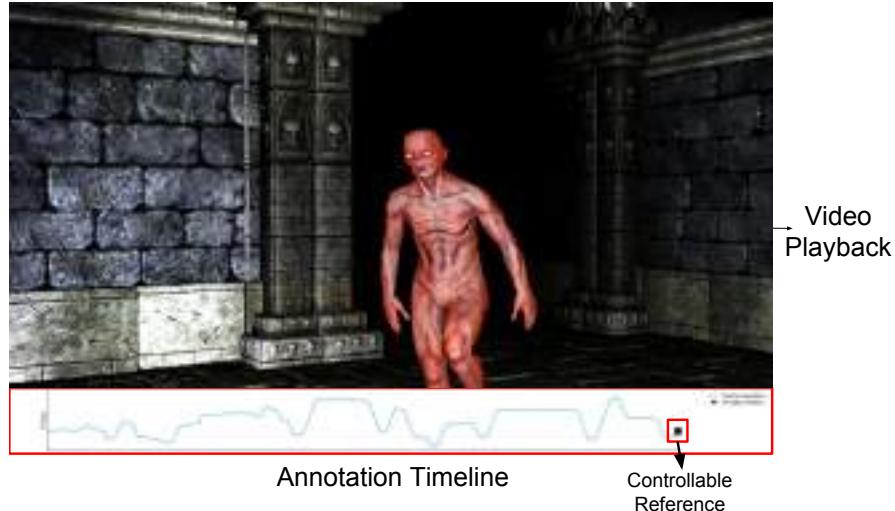


(b) *AffectRank*: A time-discrete annotation tool for arousal and valence.

(c) *FeelTrace*. A time-continuous annotation tool for arousal and valence.

**Fig. 5.5** An example of third-person annotation based on videos of players and their gameplay using either (a) the *AffectRank* (b) or the *FeelTrace* (c) annotation tool. *AffectRank* is freely available at: <https://github.com/TAPeri/AffectRank>. *FeelTrace* is freely available at: <http://emotion-research.net/download/Feeltrace%20Package.zip>.

a player's state prior to playing a game. This state can be influenced by a number of factors such as the mood of the day, the social network activity, the caffeine consumption, earlier playing activity and so on. This is a wealth of information that can be used to enrich our models. Again, what is worth detecting is the relative change [765] from the baseline state of the user prior playing to the game.



**Fig. 5.6** The *RankTrace* annotation tool. In this example the tool is used for the annotation of tension in horror games. Participants play a game and then they annotate the level of tension by watching a video-recorded playthrough of their game session (top of image). The annotation trace is controlled via a wheel-like user interface. The entire annotation trace is shown for the participant's own reference (bottom of image). Image adapted from [397]. The *RankTrace* tool is available at: <http://www.autogamedesign.eu/software>.

A **during**-experience protocol, on the other hand, may involve the player in a first-person think-aloud setup [555] or a third-person annotation design. For the latter protocol you may think of user experience experts that observe and annotate player experience during the beta release of a game, for example. As mentioned earlier, first-person annotation during play is a rather intrusive process that disrupts the gameplay and risks adding experimental noise to annotation data. In contrast, third-person annotation is not intrusive; however, there are expected deviations from the actual first-person experience, which is inaccessible to the observer.

The most popular approach for the annotation of player experience is after a game (or a series of games) has been played, in a **post**-experience fashion. Post-experience annotation is unobtrusive for the player and it is usually performed by the players themselves. Self-reports, however, are memory-dependent by nature and memory is, in turn, time-dependent. Thus, one needs to consider carefully the **time window** between the experience and its corresponding report. For the reported post-experience to be a good approximation of the actual experience the playing time window needs to be small in order to minimize memory biases, yet sufficiently large to elicit particular experiences to the player. The higher the cognitive load required to retrieve the gameplay context is, the more the reports are memory-biased and not relevant to the actual experience. Further, the longer the time window between the real experience and the self-report the more the annotator activates aspects of *episodic* memory associated with the gameplay [571]. Episodic memory traces that form the basis of

self-reports fade over time, but the precise rate at which this memory decay occurs is unknown and most likely individual [571]. Ideally, memory decay is so slow that the annotator will have a clear feeling of the gameplay session when annotating it. Now, if the time window becomes substantial—on the scale of hours and days—the annotator has to activate aspects of *semantic* memory such as general beliefs about a game. In summary, the more the episodic memory, and even more so the semantic memory, are activated during annotation, the more systematic errors are induced within the annotation data.

As a general rule of thumb the longer it takes for us to evaluate an experience of ours the larger the discrepancy between the true experience and the evaluation of the experience, which is usually more intense than the true experience. It also seems that this gap between our memory of experience and our real experience is more prominent when we report unpleasant emotions such as anger, sadness and tension rather than positive emotions [462]. Another bias that affects how we report our experience is the experience felt near the end of a session, a game level or a game; this effect has been named *peak-end rule* [462].

An effective way to assist episodic memory and minimize post-experience cognitive load is to show annotators replay videos of their gameplay (or the gameplay of others) and ask them to annotate those. This can be achieved via crowdsourcing [96] in a third-person manner or on a first-person annotation basis [125, 271, 397, 97]. Another notable approach in this direction is the *data-driven retrospective interviewing* method [187]. According to that method player behavioral data is collected and is analyzed to drive the construction of interview questions. These questions are then used in retrospect (post-experience) to reflect on the annotator’s behavior.

#### 5.5.2.6 Which Annotation Type?

We often are uncertain about the type of labels we wish to assign to a player state or a player experience. In particular, we can select from three data types for our annotation: ratings, classes, and ranks (see Fig. 5.1). The **rating**-based format represents a player’s state with a scalar value or a vector of values. Ratings are arguably the dominant practice for quantitatively assessing aspects of a user’s behavior, experience, opinion or emotion. In fact, the vast majority of user and psychometric studies have adopted rating questionnaires to capture the opinions, preferences and perceived experiences of experiment participants—see [78, 442, 119] among many. The most popular rating-based questionnaire follows the principles of a Likert scale [384] in which users are asked to specify their level of agreement with (or disagreement against) a given statement—see Fig. 5.7(a) for an example. Other popular rating-based questionnaires for user and player experience annotation include the Geneva Wheel model [600], the Self-Assessment Manikin [468], the Positive and Negative Affect Schedule [646], the Game Experience Questionnaire [286], the Flow State Scale [293] and the Player Experience of Need Satisfaction (PENS) survey [583], which was developed based on self-determination theory [162].

Rating-based reporting has notable inherent **limitations** that are often overlooked, resulting in fundamentally flawed analyses [778, 298]. First, ratings are analyzed traditionally by comparing their values across participants; see [233, 427] among many. While this is a generally accepted and dominant practice it neglects the existence of **inter-personal differences** as the meaning of each level on a rating scale may differ across experiment participants. For example, two participants assessing the difficulty of a level may assess it as exactly the same difficulty, but then one rates it as “very easy to play” and the other as “extremely easy to play”. It turns out that there are numerous factors that contribute to the different internal rating scales existent across participants [455] such as differences in personality, culture [643], temperament and interests [740]. Further, a large volume of studies has also identified the presence of primacy and recency order effects in rating-based questionnaires (e.g., [113, 773]), systematic biases towards parts of the scale [388] (e.g., right-handed participants may tend to use the right side of the scale) or a fixed tendency over time (e.g., on a series of experimental conditions, the last ones are rated higher). Indicatively, the comparative study of [773] between ratings and ranks showcases higher inconsistency effects and significant order (recency) effects existent in ratings.

In addition to inter-personal differences, a critical limitation arises when ratings are treated as interval values since ratings are by nature **ordinal values** [657, 298]. Strictly speaking, any approach or method that treats ratings as numbers by, for instance, averaging their ordinal labels is *fundamentally flawed*. In most questionnaires Likert items are represented as pictures (e.g., different representations of arousal in the Self-Assessment Manikin [468]) or as adjectives (e.g., “moderately”, “fairly” and “extremely”). These labels (images or adjectives) are often erroneously converted to integer numbers, violating basic axioms of statistics which suggest that ordinal values cannot be treated as interval values [657] since the underlying numerical scale is unknown. Note that even when a questionnaire features ratings as numbers (e.g., see Fig. 5.7(a)), the scale is still ordinal as the numbers in the instrument represent labels. Thus, the underlying numerical scale is still unknown and dependent on the participant [657, 515, 361]. Treating ratings as interval values is grounded in the assumption that the difference between consecutive ratings is fixed and equal. However, there is no valid assumption suggesting that a subjective rating scale is linear [298]. For instance, the difference between “fairly (4)” and “extremely (5)” may be larger than the distance between “moderately (3)” and “fairly (4)” as some experiment participants rarely use the extremes of the scale or tend to use one extreme more than the other [361]. If, instead, ratings are treated naturally as ordinal data no assumptions are made about the distance between rating labels, which eliminates introducing data noise to the analysis.

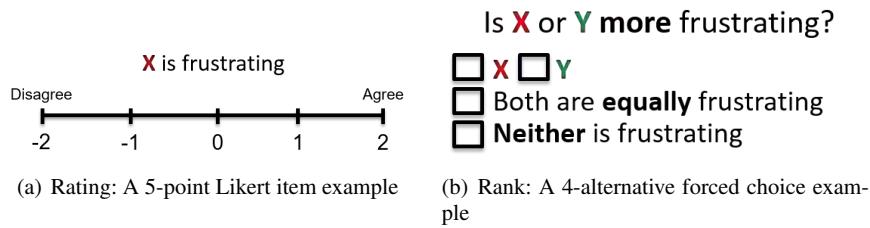
The second data type for the annotation of players is the **class-based** format. Classes allow annotators to select from a finite and non-structured set of options and, thus, a class-based questionnaire provides nominal data among two (binary) or more options. The questionnaire asks subjects to pick a player state from a particular representation which could vary from a simple boolean question (*was that game level frustrating or not? is this a sad facial expression? which level was the most*

*stressful?*) to a player state selection from, for instance, the circumplex model of affect (*is this a high- or a low-arousal game state for the player?*). The limitations of ratings are mitigated, in part, via the use of class-based questionnaires. By not providing information about the intensity of each player state, however, classes do not have the level of granularity ratings naturally have. A class-based questionnaire might also yield annotations with an unbalanced number of samples per class. A common practice in psychometrics consists of transforming sets of consecutive ratings into separate classes (e.g., see [226, 260] among many). In an example study [255], arousal ratings on a 7-point scale are transformed into *high*, *neutral* and *low* arousal classes using 7-5, 4 and 3-1 ratings, respectively. While doing so might seem appropriate, the ordinal relation among classes is not being taken into account. More importantly, the transformation process adds a new set of bias to the subjectivity bias of ratings, namely **class splitting criteria** [436].

Finally, **rank-based** questionnaires ask the annotator to rank a preference among options such as two or more sessions of the game [763]. In its simplest form, the annotator compares two options and specifies which one is preferred under a given statement (**pairwise preference**). With more than two options, the participants are asked to provide a ranking of some or all the options. Examples of rank-based questions include: *was that level more engaging than this level? which facial expression looks happier?*). Another example of a rank-based questionnaire (4-alternative forced choice) is illustrated in Fig. 5.7(b). Being a form of subjective reporting, rank-based questionnaires (as much as rating-based and class-based questionnaires) are associated with the well known limitations of memory biases and self-deception. Reporting about subjective constructs such as experience, preference or emotion via rank-based questionnaires, however, has recently attracted the interest of researchers in marketing [167], psychology [72], user modeling [761, 37] and affective computing [765, 721, 436, 455, 773] among other fields. This gradual paradigm shift is driven by both the reported benefits of ranks minimizing the effects of self-reporting subjectivity biases and recent findings demonstrating the advantages of ordinal annotation [765, 773, 455].

#### 5.5.2.7 What Is the Value of Player Experience?

Describing, labeling and assigning values to subjective notions, such as player experience, is a non-trivial task as evidenced by a number of disciplines including neuroscience [607], psychology [258], economics [630], and artificial intelligence [315]. Annotators can attempt to assign numbers to such notions in an **absolute** manner, using for instance a rating scale. Annotators can alternatively assign values in a **relative** fashion, using for instance a ranking. There are, however, a multitude of theoretical and practical reasons to doubt that subjective notions can be encoded as numbers in the first place [765]. For instance, according to Kahneman [317], co-founder of behavioral economics, “...it is safe to assume that changes are more accessible than absolute values”; his theory about judgment heuristics is built on Herbert Simon’s *psychology of bounded rationality* [630]. Further, an important



**Fig. 5.7** Examples of rating-based (a) vs. rank-based (b) questionnaires.

thesis in psychology, named **adaptation level theory** [258], suggests that humans lack the ability to maintain a constant value about subjective notions and their preferences about options are, instead, made on a pairwise comparison basis using an internal ordinal scale [460]. The thesis claims that while we are efficient at discriminating among options, we are not good at assigning accurate absolute values for the intensity of what we perceive. For example, we are particularly bad at assigning absolute values to tension, frequency and loudness of sounds, the brightness of an image, or the arousal level of a video. The above theories have also been supported by neuroscientific evidence suggesting that experience with stimuli gradually creates our own internal context, or *anchor* [607], against which we rank any forthcoming stimulus or perceived experience. Thus, our choice about an option is driven by our internal ordinal representation of that particular option within a sample of options; not by any absolute value of that option [658].

As a remote observation, one may argue that the relative assessment provides less information than the absolute assessment since it does not express a quantity explicitly and only provides ordinal relations. As argued earlier, however, any additional information obtained in an absolute fashion (e.g., when ratings are treated as numbers) violates basic axioms of applied statistics. Thus the value of the additional information obtained (if any) is questioned directly [765].

In summary, results across different domains investigating subjective assessment suggest that relative (rank-based) annotations minimize the assumptions made about experiment participants' notions of highly subjective constructs such as player experience. Further, annotating experience in a relative fashion, instead of an absolute fashion, leads to the construction of more generalizable and accurate computational models of experience [765, 436].

### 5.5.3 No Output

Very often we are faced with datasets where target outputs about player behavioral or experience states are not available. In such instances modeling of players must rely on unsupervised learning [176, 244, 178] (see Fig. 5.1). Unsupervised learning,

as discussed in Chapter 2, focuses on fitting a model to observations by discovering associations of the input and without having access to a target output. The input is generally treated as a set of random variables and a model is built through the observations of associations among the input vectors. Unsupervised learning as applied to modeling players involves tasks such as **clustering** and **association mining** which are described in Section 5.6.3.

It may also be the case that we do not have target outputs available but, nevertheless, we can design a **reward** function that characterizes behavioral or experiential patterns of play. In such instances we can use reinforcement learning approaches to discover policies about player behavior or player experience based on in-game play traces or other state-action representations (see Section 5.6.2). In the following section we detail the approaches used for modeling players in a supervised learning, reinforcement learning and unsupervised learning fashion.

## 5.6 How Can We Model Players?

In this section we build upon the data-driven approach of player modeling and discuss the application of **supervised**, **reinforcement** and **unsupervised** learning to model players, their behavior and their experience. To showcase the difference between the three learning approaches let us suppose we wish to classify player behavior. We can only use unsupervised learning if no behavioral classes have been defined a priori [176]. We can instead use supervised learning if, for example, we have already obtained an initial classification of players (either manually or via clustering) and we wish to fit new players into these predefined classes [178]. Finally, we can use reinforcement learning to derive policies that imitate different types of playing behavior or style. In Section 5.6.1 we focus on the supervised learning paradigm whereas in Section 5.6.2 and Section 5.6.3 we outline, respectively, the reinforcement learning and the unsupervised learning approach for modeling players. All three machine learning approaches are discussed in Chapter 2.

### 5.6.1 Supervised Learning

Player modeling consists of finding a function that maps a set of measurable attributes of the player to a particular player state. Following the supervised learning approach this is achieved by machine learning, or automatically adjusting, the parameters of a model to fit a dataset that contains a set of input samples, each one paired with target outputs. The input samples correspond to the list of measurable attributes (or features) while the target outputs correspond to the annotations of the player's states for each of the input samples that we are interested to learn to predict. As mentioned already, the annotations may vary from behavioral characteris-

tics, such as completion times of a level or player archetypes, to estimates of player experience, such as player frustration.

As we saw in Chapter 2 popular supervised learning techniques, including artificial neural networks (shallow or deep architectures), decision trees, and support vector machines, can be used in games for the analysis, the imitation and the prediction of player behavior, and the modeling of playing experience. The data type of the annotation determines the output of the model and, in turn, the type of the machine learning approach that can be applied. The three supervised learning alternatives for learning from numerical (or interval), nominal and ordinal annotations—respectively, **regression**, **classification** and **preference learning**—are discussed in this section.

### 5.6.1.1 Regression

When the outputs that a player model needs to approximate are interval values, the modeling problem is known as metric or standard **regression**. Any regression algorithm is applicable to the task, including linear or polynomial regression, artificial neural networks and support vector machines. We refer the reader to Chapter 2 for details on a number of popular regression algorithms.

Regression algorithms are appropriate for **imitation** and **prediction** tasks of player *behavior*. When the task, however, is modeling of player *experience* caution needs to be put on the data analysis. While it is possible, for instance, to use regression algorithms to learn the exact numeric ratings of experience, in general it should be avoided because regression methods assume that the target values follow an interval (numerical) scale. Ratings naturally define an ordinal scale [765, 773] instead. As mentioned already, ordinal scales such as ratings should not be converted to numerical values due to the subjectivity inherent to reports, which imposes a non-uniform and varying distance among questionnaire items [778, 657]. Prediction models trained to approximate a real-value representation of a rating—even though they may achieve high prediction accuracies—do not necessarily capture the true reported playing experience because the ground truth used for training and validation of the model has been undermined by the numerous effects discussed above. We argue that the known fundamental pitfalls of self-reporting outlined above provide sufficient evidence against the use of regression for player experience modeling [765, 515, 455, 361]. Thus, we leave the evaluation of regression methods on experience annotations outside the scope of this book.

### 5.6.1.2 Classification

**Classification** is the appropriate form of supervised learning for player modeling when the annotation values represent a finite and non-structured set of classes. Classification methods can infer a mapping between those classes and player attributes. Available algorithms include artificial neural networks, decision trees, ran-

dom forests, support vector machines,  $K$ -nearest neighbors, and ensemble learning among many others. Further details about some of these algorithms can be found in Chapter 2.

Classes can represent playing behavior which needs to be **imitated** or , such as completion times (e.g., expressed as low, average or high completion time) or user retention in a free-to-play game (e.g., expressed as weak, mild or strong retention). Classes can alternatively represent player experience such as an *excited* versus a *frustrated* player as manifested from facial expressions or *low*, *neutral* and *high* arousal states for a player.

Classification is perfectly suited for the task of modeling player experience if discrete annotations of experience are selected from a list of possibilities and provided as target outputs [153, 344]. In other words, annotations of player experience need to be nominal for classification to be applied. A common practice, however, as already mentioned in Section 5.5.2.6, is to treat ratings of experience as classes and transform the ordinal scale—that defines ratings—into a nominal scale of separate classes. For example, ratings of arousal that lie between  $-1$  and  $1$  are transformed into low, neutral and high arousal classes. By classifying ratings not only the ordinal relation among the introduced classes is ignored but, most importantly, the transformation process induces several biases to the data (see Section 5.5.2.6). These biases appear to be detrimental and mislead the search towards the ground truth of player experience [765, 436].

### 5.6.1.3 Preference Learning

As an alternative to regression and classification methods, **preference learning** [215] methods are designed to learn from ordinal data such as ranks or preferences. It is important to note that the training signal in the preference learning paradigm merely provides information for the **relative** relation between instances of the phenomenon we attempt to approximate. Target outputs that follow an ordinal scale do not provide information about the intensity (regression) or the clusters (classification) of the phenomenon.

Generally we could construct a player model based on in-game *behavioral preferences*. The information that this player, for example, prefers the mini-gun over a number of other weapons could form a set of pairwise preferences we could learn from. Alternatively we can build a model based on *experience preferences*. A player, for instance, reported that area X of the level is more challenging than area Y of the same level. Based on a set of such pairwise preferences we can derive a global function of challenge for that player.

As outlined in Chapter 2 a large palette of algorithms is available for the task of preference learning. Many popular classification and regression techniques have been adapted to tackle preference learning tasks, including linear statistical models such as linear discriminant analysis and large margins, and non-linear approaches such as Gaussian processes [122], deep and shallow artificial neural networks [430], and support vector machines [302].

Preference learning has already been extensively applied to modeling aspects of players. For example, Martínez et al. [430, 431] and Yannakakis et al. [780, 771] have explored several artificial neural network approaches to learn to predict affective and cognitive states of players reported as pairwise preferences. Similarly, Garbarino et al. [217] have used linear discriminant analysis to learn pairwise enjoyment predictors in racing games. To facilitate the use of proper machine learning methods on preference learning problems, a number of such preference learning methods as well as data preprocessing and feature selection algorithms have been made available as part of the Preference Learning Toolbox (PLT) [198]. PLT is an open-access, user-friendly and accessible toolkit<sup>5</sup> built and constantly updated for the purpose of easing the processing of (and promoting the use of) ranks.

As ratings, by definition, express ordinal scales they can directly be transposed to any ordinal representation (e.g., pairwise preferences). For instance, given an annotator's rating indicating that a condition A felt 'slightly frustrating' and a condition B felt 'very frustrating', a preference learning method can train a model that predicts a higher level of frustration for B than for A. In this way the modeling approach avoids introducing artifacts of what is the actual difference between 'very' and 'slightly' or the usage of the scale for this particular annotator. Further, the limitation of different subjective scales across users can be safely bypassed by transforming rating reports into ordinal relations on a per-annotator basis. Finally, the problem of the scale varying across time due to episodic memory still persists but can be minimized by transforming only consecutive reports, i.e., given a report for three conditions A, B and C, the player model can be trained using only the relation between A and B, and B and C (dropping the comparison between A and C).

#### 5.6.1.4 Summary: The Good, the Bad and the Ugly

The last section on supervised learning is dedicated to the comparison among the three methods—regression, classification and preference learning—for modeling players. Arguably the discussion is limited when the in-game *behavior* of players is imitated or predicted. If behavioral data about players follows interval, nominal or ordinal scales then naturally, regression, classification and preference learning should be applied, respectively. Behavioral data have an objective nature which makes the task of learning less challenging. Given the subjective notion of player *experience*, however, there are a number of caveats and limitations of each algorithm that need to be taken into account. Below we discuss the comparative advantages of each and we summarize the key outcomes of supervised learning as applied to modeling the experience of players.

**Regression vs. Preference Learning:** Motivated by psychological studies suggesting that interval ratings misrepresent experience [515, 455, 361], we will not dedicate ourselves an extensive comparison between preference learning and regression methods. The performance comparison between a regression and preference-

---

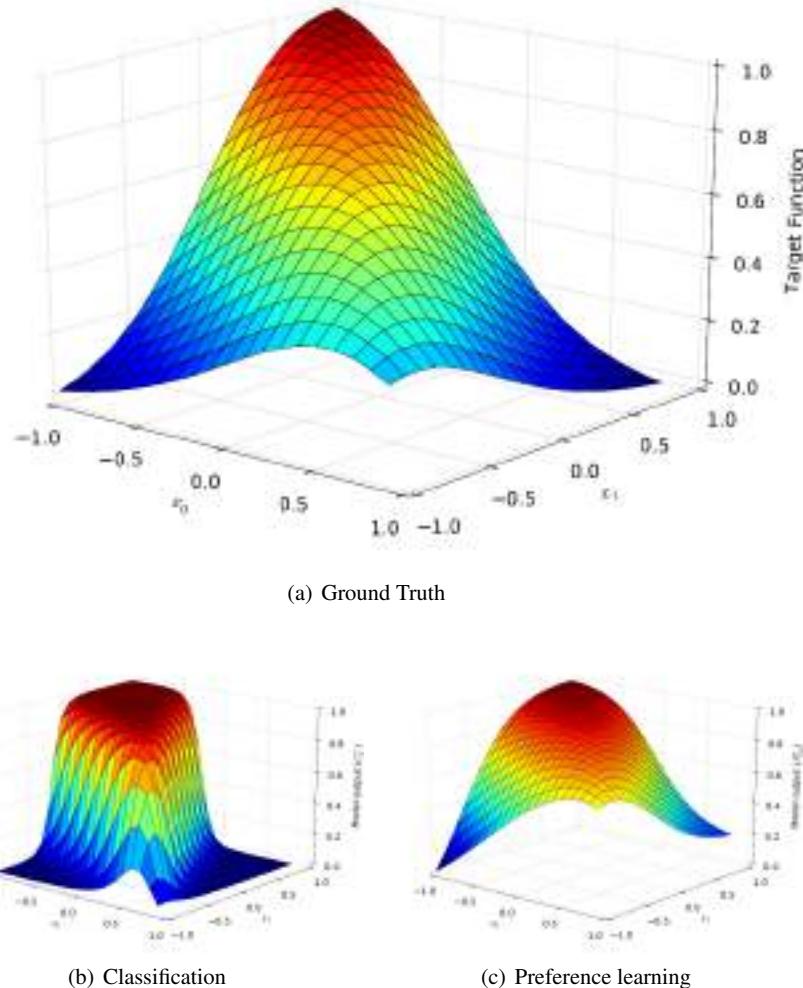
<sup>5</sup> <http://sourceforge.net/projects/pl-toolbox/>

learned model is also irrelevant as the former is arguably *a priori* incapable of capturing the underlying experience phenomenon as precisely as the latter. Such deviations from the ground truth, however, are not trivial to illustrate through a data modeling approach and thus the comparison is not straightforward. The main reason is that the objective ground truth is fundamentally ill-defined when numbers are used to characterize subjective notions such as player experience.

**Regression vs. Classification:** Classes are easy to analyze and create player models from. Further, their use eliminates part of the inter-personal biases introduced with ratings. For these reasons classification should be preferred to regression for player experience modeling. We already saw that classification, instead of regression, is applied when ratings are available to overcome part of the limitations inherent in rating-based annotation. For instance, this can be achieved by transforming arousal ratings to *high*, *neutral* and *low* arousal classes [255]. While this common practice in psychometrics eliminates part of the rating subjectivity it adds new forms of data biases inherent in the ad-hoc decisions to split the classes. Further, the analysis of player models across several case studies in the literature has already shown that transforming ratings into classes creates a more complicated machine learning problem [765, 436].

**Classification vs. Preference Learning:** Preference learning is the supreme method for modeling experience when ranks or pairwise preferences are available. Even when ratings or classes are available comparisons between classification and preference learning player models in the literature suggest that preference learning methods lead to more efficient, generic and robust models which capture more information about the ground truth [765]. Indicatively, Crammer and Singer [147] compare classification, regression and preference learning training algorithms in a task to learn ratings. They report the supremacy of preference learning over the other methods based on several synthetic datasets and a movie-ratings dataset. In addition, extensive evidence already shows that preference learning better approximates the underlying function between input (e.g., experience manifestations such as gameplay) and output (e.g., annotations) [436]. Figure 5.8 showcases how much closer a preference learned model can reach a hypothesized (artificial) ground truth, compared to a classification model trained on an artificial dataset. In summary, preference learning via rank-based annotation controls for reporting memory effects, eliminates subjectivity biases and builds models that are closer to the ground truth of player experience [778, 777].

Grounded in extensive evidence our final note for the selection of a supervised learning approach for modeling player experience is clear: Independently of how experience is annotated we argue that preference learning (*the good*) is a superior supervised learning method for the task at hand, classification (*the ugly*) provides a good balance between simplicity and approximation of the ground truth of player experience whereas regression (*the bad*) is based on rating annotations which are of questionable quality with respect to their relevance to the true experience.



**Fig. 5.8** A hypothesized (artificial) ground truth function (z-axis) which is dependent on two player attributes,  $x_1$  and  $x_2$  (Fig. 5.8(a)), the best classification model (Fig. 5.8(b)) and the best preference learned model (Fig. 5.8(c)). All images are retrieved from [436].

### 5.6.2 Reinforcement Learning

While it is possible to use **reinforcement learning** to model aspects of users during their interaction the RL approach for modeling players has been tried mostly in comparatively simplistic and abstract games [195] and has not seen much applica-

tion in computer games. The key motivation for the use of RL for modeling players is that it can capture the relative valuation of game states as encoded internally by humans during play [685]. At first glance, player modeling with RL may seem to be an application of RL for game playing, and we discuss this in Chapter 3 as part of the *play for experience* aim. In this section, we instead discuss this approach from the perspective that a policy learned via RL can capture internal player states with no corresponding absolute target values such as decision making, learnability, cognitive skills or emotive patterns. Further, those policies can be trained on player data such as **play traces**. The derived player model depicts psychometrically-valid, abstract simulations of a human player’s internal cognitive or affective processes. The model can be used directly to interpret human play, or indirectly, it can be featured in AI agents which can be used as playtesting bots during the game design process, as baselines for adapting agents to mimic classes of human players, or as believable, human-like opponents [268].

Using the RL paradigm, we can construct player models via RL if a reward signal can adjust a set of parameters that characterize the player. The reward signal can be based either directly on the in-game behavior of the player—for instance, the decision taken at a particular game state—or indirectly on player annotations (e.g., annotated excitement throughout the level) or objective data (e.g., physiological indexes showing player stress). In other words, the immediate reward function can be based on gameplay data if the model wishes to predict the behavior of the player or, instead, be based on any objective measure or subjective report if the model attempts to predict the experience of the player. The representation of the RL approach can be anything from a standard Q table that, for instance, models the decision making behavior of a player (e.g., as in [268, 685]) to an ANN that e.g., models in-game behavior (as in [267]), to a set of behavior scripts [650] that are adjusted to imitate gameplay behavior via RL, to a deep Q network.

We can view two ways of constructing models of players via RL: models can be built **offline** (i.e., before gameplay starts) or at **runtime** (i.e., during play). We can also envision hybrid approaches by which models are first built offline and then are polished at runtime. Offline RL-based modeling adds value to our playtesting capacity via, for instance, *procedural personas* (see Section 5.7.1.3) whereas runtime RL-based player modeling offers *dynamicity* to the model with respect to time. Runtime player modeling further adds on the capacity of the model to adapt to the particular characteristics of the player, thereby increasing the degree of personalization. We can think of models of players, for instance, that are continuously tailored to the current player by using the player’s in-game annotations, behavioral decisions, or even physiological responses during the game.

This way of modeling players is still in its infancy with only a few studies existent on player behavioral modeling in educational games [488], in roguelike adventure games [268, 267] via TD learning or evolutionary reinforcement learning, and in first-person shooter games [685] via inverse RL. However, the application of RL for modeling users beyond games has been quite active for the purposes of modeling web-usage data and interactions on the web [684] or modeling user simulations in dialog systems [114, 225, 595]. Normally in such systems a statistical model is

first trained on a corpus of human-computer interaction data for simulating (imitating) user behavior. Then reinforcement learning is used to tailor the model towards an optimal dialog strategy which can be found through trial and error interactions between the user and the simulated user.

### 5.6.3 Unsupervised Learning

The aim of **unsupervised learning** (see Chapter 2) is to derive a model given a number of observations. Unlike in supervised learning, there is no specified target output. Unlike in reinforcement learning there is no reward signal. (In short, there is no training signal of any kind.) In unsupervised learning, the signal is hidden internally in the interconnections among data attributes. So far, unsupervised learning has mainly been applied to two core player modeling tasks: **clustering** behaviors and **mining associations** between player attributes. While Chapter 2 provides the general description of these unsupervised learning algorithms, in this section we focus on their specific application for modeling players.

#### 5.6.3.1 Clustering

As discussed in Chapter 2, **clustering** is a form unsupervised learning aiming to find clusters in datasets so that data within a cluster is similar to each other and dissimilar to data in other clusters. When it comes to the analysis of user behavior in games, clustering offers a means for reducing the dimensionality of a dataset, thereby yielding a manageable number of critical features that represent user behavior. Relevant data for clustering in games include player behavior, navigation patterns, assets bought, items used, game genres played and so on. Clustering can be used to group players into archetypical playing patterns in an effort to evaluate how people play a particular game and as part of a user-oriented testing process [176]. Further, one of the key questions of user testing in games is *whether people play the game as intended*. Clustering can be utilized to derive a number of different playing or behavioral styles directly addressing this question. Arguably, the key challenge in successfully applying clustering in games is that the derived clusters should have an intelligible meaning with respect to the game in question. Thus, clusters should be clearly interpretable and labeled in a language that is meaningful to the involved stakeholders (such as designers, artists and managers) [176, 178]. In the case studies of Section 5.7.1 we meet the above challenges and demonstrate the use of clustering in the popular *Tomb Raider: Underworld* (EIDOS interactive, 2008) game.

### 5.6.3.2 Frequent Pattern Mining

In Chapter 2 we defined frequent pattern mining as the set of problems and techniques related to finding patterns and structures in data. Patterns include sequences and itemsets. Both **frequent itemset mining** (e.g., Apriori [6]) and **frequent sequence mining** (e.g., GSP [652]) are relevant and useful for player modeling. The key motivation for applying frequent pattern mining on game data is to find inherent and hidden regularities in the data. In that regard, key player modeling problems, such as player type identification and detection of player behavior patterns, can be viewed as frequent pattern mining problems. Frequent pattern mining can for example be used to discover what game content is often purchased together—e.g., players that buy X tend to buy Y too—or what are the subsequent actions after dying in a level—e.g., players that die often in the tutorial level pick up more health packs in level 1 [120, 621].

## 5.7 What Can We Model?

As already outlined at the beginning of this chapter, modeling of users in games can be classified into two main tasks: modeling of the **behavior** of players and modeling of their **experience**. It is important to remember that modeling of player behavior is (mostly) a task of **objective** nature whereas the modeling of player experience is **subjective** given the idiosyncratic nature of playing experience. The examples we present in the remainder of this chapter highlight the various uses of AI for modeling players.

### 5.7.1 Player Behavior

In this section, we exemplify player behavior modeling via three representative use cases. The two first examples are based on a series of studies on player modeling by Drachen et al. in 2009 [176] and later on by Mahlmann et al. in 2010 [414] in the *Tomb Raider: Underworld* (EIDOS interactive, 2008) game. The analysis includes both the **clustering** of players [176] and the **prediction** [414] of their behavior, which make it an ideal case study for the purposes of this book. The third study presented in this section focuses on the use of play traces for the procedural creation of player models. That case study explores the creation of **procedural personas** in the *MiniDungeons* puzzle roguelike game.



**Fig. 5.9** A screenshot from the *Tomb Raider: Underworld* (EIDOS interactive, 2008) game featuring the player character, Lara Croft. The game is used as one of the player behavior modeling case studies presented in this book. Image obtained from Wikipedia (fair use).

#### 5.7.1.1 Clustering Players in *Tomb Raider: Underworld*

*Tomb Raider: Underworld* (TRU) is a third-person perspective, advanced platform-puzzle game, where the player has to combine strategic thinking in planning the 3D-movements of Lara Croft (the game's player character) and problem solving in order to go through a series of puzzles and navigate through a number of levels (see Fig. 5.9).

The **dataset** used for this study includes entries from 25,240 players. The 1,365 of those that completed the game were selected and used for the analysis presented below. Note that TRU consists of seven main levels plus a tutorial level. Six features of gameplay behavior were extracted from the data and are as follows: number of deaths by opponents, number of deaths by the environment (e.g., fire, traps, etc.), number of deaths by falling (e.g., from ledges), total number of deaths, game completion time, and the times help was requested. All six features were calculated on the basis of completed TRU games. The selection of these particular features was based on the core game design of the TRU game and their potential impact on the process of distinguishing among dissimilar patterns of play.

Three different **clustering** techniques were applied to the task of identifying the number of meaningful and interpretable clusters of players in the data: k-means, hierarchical clustering and self-organizing maps. While the first two have been covered in Chapter 2 we will briefly outline the third method here.

A **self-organizing map** (SOM) [347] creates and iteratively adjusts a low-dimensional projection of the input space via vector quantization. In particular, a type of large SOM called an emergent self-organizing map [727] was used in conjunction with reliable visualization techniques to help us identify clusters. A SOM consists of neurons organized in a low-dimensional grid. Each neuron in the grid

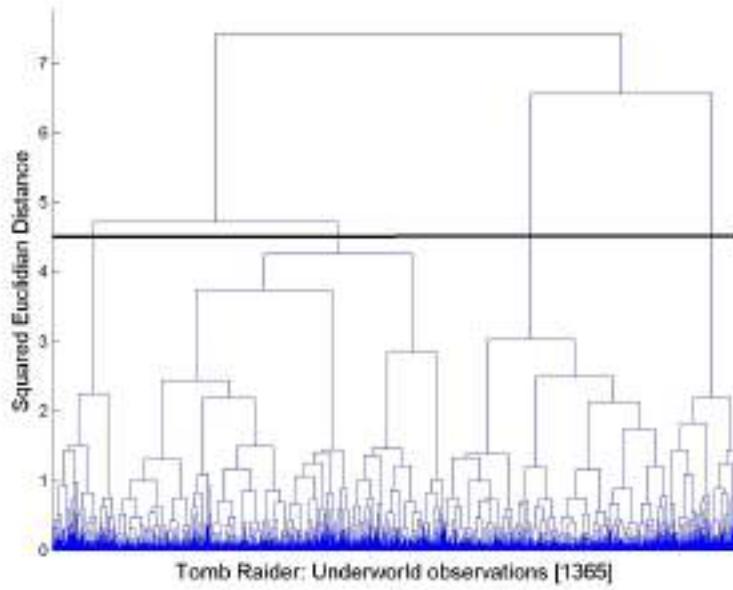
(map) is connected to the input vector through a connection weight vector. In addition to the input vector, the neurons are connected to neighbor neurons of the map through neighborhood interconnections which generate the structure of the map: rectangular and hexagonal lattices organized in a two-dimensional sheet or a three-dimensional toroid shape are some of the most popular topologies used. SOM training can be viewed as a **vector quantization** algorithm which resembles the k-means algorithm. What differentiates SOM, however, is the update of the topological neighbors of the best-matching neuron—a best-matching neuron is a neuron for which there exists at least one input vector for which the Euclidean distance to the weight vector of this neuron is minimal. As a result, the whole neuron neighborhood is stretched towards the presented input vector. The outcome of SOM training is that neighboring neurons have similar weight vectors which can be used for projecting the input data to the two-dimensional space and thereby clustering a set of data through observation on a 2D plane. For a more detailed description of SOMs, the reader is referred to [347].

To get some insight into the possible number of clusters existent in the data, **k-means** was applied for all  $k$  values less than or equal to 20. The sum of the Euclidean distances between each player instance and its corresponding cluster centroid (i.e., quantization error) is calculated for all 20 trials of k-means. The analysis reveals that the percent decrease of the mean quantization error due to the increase of  $k$  is notably high when  $k = 3$  and  $k = 4$ . For  $k = 3$  and  $k = 4$  this value equals 19% and 13% respectively while it lies between 7% and 2% for  $k > 4$ . Thus, the k-means clustering analysis provides the first indication of the existence of three or four main player behavioral clusters within the data.

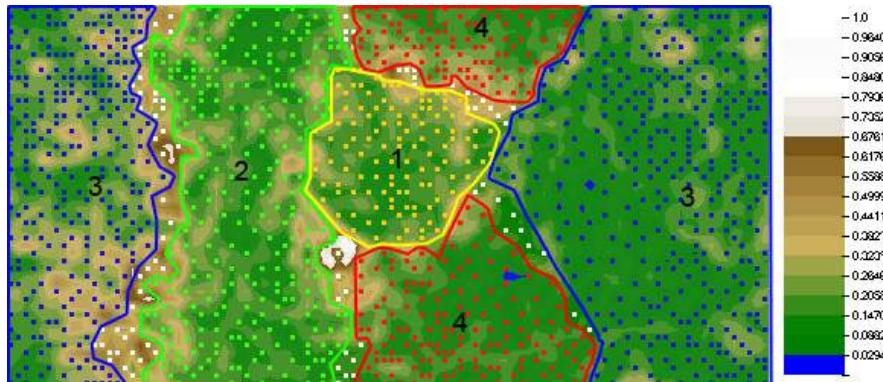
As an alternative approach to k-means, **hierarchical clustering** is also applied to the dataset. This approach seeks to build a hierarchy of clusters existent in the data. The Ward's clustering method [747] is used to specify the clusters in the data by which the squared Euclidean distance is used as a measure of dissimilarity between data vector pairs. The resulting **dendrogram** is depicted in Fig. 5.10(a). As noted in Chapter 2 a dendrogram is a treelike diagram that illustrates the merging of data into clusters as a result of hierarchical clustering. It consists of many U-shaped lines connecting the clusters. The height of each U represents the squared Euclidean distance between the two clusters being connected. Depending on where the data analyst sets the squared Euclidean distance threshold a dissimilar number of clusters can be observed.

Both k-means and hierarchical clustering already demonstrate that the 1,365 players can be clustered in a low number of different player types. K-means indicates there are for three or four clusters while the Ward's dendrogram reveals the existence of two populated and two smaller clusters, respectively, in the middle and at the edges of the tree.

Applying SOM, as the third alternative approach, allows us to cluster the TRU data by observation on a two-dimensional plane. The U-matrix depicted in Fig. 5.10(b) is a visualization of the local distance structure in the data placed onto a two-dimensional map. The average distance value between each neuron's weight vector and the weight vectors of its immediate neighbors corresponds to the height



(a) Dendrogram of TRU data using Ward hierarchical clustering. A squared Euclidean distance of 4.5 (illustrated with a horizontal black line) reveals four clusters. Image adapted from [176].



(b) U-matrix visualization of a self-organizing map depicting the four player clusters identified in a population of 1,365 TRU players (shown as small colored squares). Different square colors depict different player clusters. Valleys represent clusters whereas mountains represent cluster borders. Image adopted from [176].

**Fig. 5.10** Detecting player types from TRU data using hierarchical (a) and SOM (b) clustering methods.

of that neuron in the U-matrix (positioned at the map coordinates of the neuron).

Thus, U-matrix values are large in areas where no or few data points reside, creating mountain ranges for cluster boundaries. On the other hand, visualized valleys indicate clusters of data as small U-matrix values are observed in areas where the data space distances of neurons are small.

The SOM analysis reveals four main classes of behavior (player types) as depicted in Fig. 5.10(b). The different colors of the U-matrix correspond to the four different clusters of players. In particular, cluster 1 (8.68% of the TRU players) corresponds to players that die very few times, their death is caused mainly by the environment, they do not request help from the game frequently and they complete the game very quickly. Given such game skills these players were labeled as *Veterans*. Cluster 2 (22.12%) corresponds to players that die quite often (mainly due to falling), they take a long time to complete the game—indicating a slow-moving, careful style of play—and prefer to solve most puzzles in the game by themselves. Players of this cluster were labeled as *Solvers*, because they excel particularly at solving the puzzles of the game. Players of cluster 3 form the largest group of TRU players (46.18%) and are labeled as *Pacifists* as they die primarily from active opponents. Finally, the group of players corresponding to cluster 4 (16.56% of TRU players), namely the *Runners*, is characterized by low completion times and frequent deaths by opponents and the environment.

The results showcase how clustering of player behavior can be useful to evaluate game designs. Specifically, TRU players seem to not merely follow a specific strategy to complete the game but rather they fully explore the affordances provided by the game in dissimilar ways. The findings are directly applicable to TRU game design as clustering provides answers to the critical question of *whether people play the game as intended*. The main limitation of clustering, however, is that the derived clusters are not intuitively interpretable and that clusters need to be represented into meaningful behavioral patterns to be useful for game design. Collaborations between the data analyst and the designers of the game—as was performed in this study—is essential for meaningful interpretations of the derived clusters. The benefit of such a collaboration is both the enhancement of game design features and the effective **phenomenological debugging** of the game [176]. In other words, we make sure both that no feature of the game is underused or misused and that the playing experience and the game balance are debugged.

### 5.7.1.2 Predicting Player Behavior in *Tomb Raider: Underworld*

Building on the same set of TRU player data, a second study examined the possibilities of **predicting** particular aspects of playing behavior via supervised learning [414]. An aspect of player behavior that is particularly important for game design is to predict *when a player will stop playing*. As one of the perennial challenges of game design is to ensure that as many different types of players are facilitated in the design, being able to predict when players will stop playing a game is of interest because it assists with locating potentially problematic aspects of game design. Fur-

ther, such information helps toward the redesign of a game's monetization strategy for maximizing user retention.

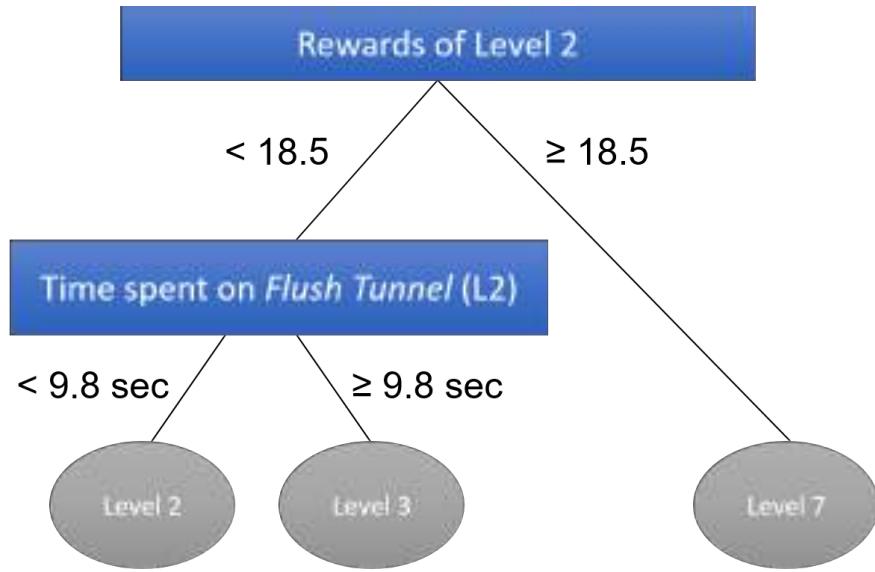
**Data** was drawn from the Square Enix Europe Metrics Suite and was collected during a two month period (1st Dec 2008 – 31st Jan 2009), providing records from approximately 203,000 players. For the player behavior prediction task it was decided to extract a subsample of 10,000 players which provides a large enough and representative dataset for the aims of the study, while at the same time is manageable in terms of computational effort. A careful data-preprocessing approach yielded 6,430 players that were considered for the prediction task—these players had completed the first level of the game.

As in the TRU cluster analysis the features extracted from the data relate to the core mechanics of the game. In addition to the six features investigated in the clustering study of TRU the extracted features for this study include the number of times the adrenalin feature of the game was used, the number of rewards collected, the number of treasures found, and the number of times the player changes settings in the game (including player ammunition, enemy hit points, player hit points, and recovery time when performing platform jumps). Further details about these features can be found in [414].

To test the possibility of predicting the TRU level the player completed last a number of **classification** algorithms are tested on the data using the Weka machine learning software [243]. The approach followed was to experiment with at least one algorithm from each of the algorithm families existent in Weka and to put additional effort on those classification algorithms that were included in a recent list of the most important algorithms in data mining: decision tree induction, backpropagation and simple regression [759]. The resulting set of algorithms chosen for classification are as follows: logistic regression, multi-layer perceptron backpropagation, variants of decision trees, Bayesian networks, and support vector machines. In the following section, we only outline the most interesting results from those reported in [414]. For all tested algorithms, the reported classification prediction accuracy was achieved through 10-fold cross validation.

Most of the tested algorithms had similar levels of performance, and were able to predict when a player will stop playing substantially better than the baseline. In particular, considering only gameplay of level 1 classification algorithms reach an accuracy between 45% and 48%, which is substantially higher than the baseline performance (39.8%). When using additional features from level 2, the predictions are much more accurate—between 50% and 78%—compared to the baseline (45.3%). In particular, decision trees and logistic regression manage to reach accuracies of almost 78% in predicting on what level a player will stop playing. The difference in the predictive strength of using level 1 and 2 data as compared to only level 1 data is partly due to increased amount of features used in the latter case.

Beyond accuracy an important feature of machine learning algorithms is their transparency and their expressiveness. The models are more useful to a data analyst and a game designer if they can be expressed in a form which is easy to visualize and comprehend. Decision trees—of the form constructed by the ID3 algorithm [544] and its many derivatives—are excellent from this perspective, especially if pruned to



**Fig. 5.11** A decision tree trained by the ID3 algorithm [544] to predict when TRU players will stop playing the game. The leaves of the tree (ovals) indicate the number of the level (2, 3 or 7) the player is expected to complete. Note that the TRU game has seven levels in total. The tree is constrained to tree depth 2 and achieves a classification accuracy of 76.7%.

a small size. For instance, the extremely small decision tree depicted in Fig. 5.11 is constrained to tree depth 2 and was derived on the set of players who completed both levels 1 and 2 with a classification accuracy of 76.7%. The predictive capacity of the decision tree illustrated in Fig. 5.11 is impressive given how extremely simple it is. The fact that we can predict the final played level—with a high accuracy—based only on the amount of time spent in the room named *Flush Tunnel* of level 2 and the total rewards collected in level 2 is very appealing for game design. What this decision tree indicates is that the amount of time players spend within a given area early in the game and how well they perform are important for determining if they continue playing the game. Time spent on a task or in an area of the game can indeed be indicative of challenges with progressing through the game, which can result in a frustrating experience.

#### 5.7.1.3 Procedural Personas in *MiniDungeons*

**Procedural personas** are generative models of player behavior, meaning that they can replicate in-game behavior and be used for playing games in the same role as players; additionally, procedural personas are meant to represent archetypical players rather than individual players [268, 267, 269]. A procedural persona can be defined as the parameters of a utility vector that describe the preferences of a

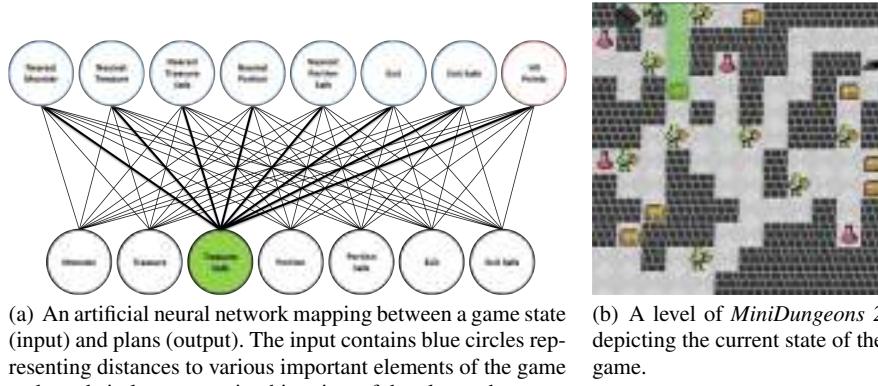
player. For example, a player might allocate different weight to finishing a game fast, exploring dialog options, getting a high score, etc.; these preferences can be numerically encoded in a utility vector where each parameter corresponds to the persona’s interest in a particular activity or outcome. Once these utilities are defined, reinforcement learning via TD learning or neuroevolution can be used to find a policy that reflects these utilities, or a tree search algorithm such as MCTS can be used with these utilities as evaluation functions. Approaches similar to the procedural persona concept have also been used for modeling the learning process of the player in educational games via reinforcement learning [488].

As outlined in Section 5.4.1, *MiniDungeons* is a simple rogue-like game which features turn-based discrete movement, deterministic mechanics and full information. The player avatar must reach the exit of each level to win it. Monsters block the way and can be destroyed, at the cost of decreasing the player character’s health; health can be restored by collecting potions. Additionally, treasures are distributed throughout levels. In many levels, potions and treasures are placed behind monsters, and monsters block the shortest path from the entrance to the exit. Like many games, it is therefore possible to play *MiniDungeons* with different goals, such as reaching the exit in the shortest possible time, collecting as much treasure as possible or killing all the monsters (see Figs. 5.3 and 5.12).

These different playing styles can be formalized as procedural personas by attaching differing utilities to measures such as the number of treasures collected, the number of monsters killed or the number of turns taken to reach the exit. Q-learning can be used to learn policies that implement the appropriate persona in single levels [268], and evolutionary algorithms can be used to train neural networks that implement a procedural persona across multiple levels [267]. These personas can be compared with the play traces of human players by placing the persona in every situation that the human encountered in the play trace and comparing the action chosen by the procedural persona with the action chosen by the human (as you are comparing human actions with those of a Q function, you might say that you are asking “what would Q do?”). It is also possible to learn the utility values for a procedural persona clone of a particular human player by evolutionary search for those values that make the persona best match a particular play trace (see Fig. 5.12). However, it appears that these “clones” of human players generalize less well than designer-specified personas [269].

### 5.7.2 Player Experience

The modeling of player experience involves learning a set of target outputs that approximate the experience (as opposed of the behavior) of the player. By definition, that which is being modeled (experience) is of **subjective nature** and the modeling therefore requires target outputs that somehow approximate the ground truth of experience. A model of player experience predicts some aspect of the experience a player would have in some game situation, and learning such models is naturally a



**Fig. 5.12** A example of a *procedural persona*: In this example we evolve the weights of artificial neural networks—an ANN per persona. The ANN takes observations of the player character and its environment and uses these to choose from a selection of possible plans. During evolution the utility function of each persona is used as the fitness function for adjusting its network weights. Each individual of each generation is evaluated by simulating a full game. A utility function allows us to evolve a network to pursue multiple goals across a range of different situations. The method depends on the designer providing carefully chosen observations, appropriate planning algorithms, and well-constructed utility functions. In this example the player opts to move towards a safe treasure. This is illustrated with a green output neuron and highlighted corresponding weights (a) and a green path on the game level (b).

supervised learning problems. As mentioned, there are many ways this can be done, with approaches to player experience modeling varying regarding the inputs (from what the experience is predicted, e.g., physiology, level design parameters, playing style or game speed), the outputs (what sort of experience is predicted, e.g., fun, frustration, attention or immersion) and the modeling methodology.

In this section we will outline a few examples of supervised learning for modeling the experience of players. To best cover the material (methods, algorithms, uses of models) we rely on studies which have been thoroughly examined in the literature. In particular, in the remainder of this section we outline the various approaches and extensive methodology for modeling experience in two games: a variant of the popular *Super Mario Bros* (Nintendo, 1985) game and a 3D prey-predator game named *Maze-Ball*.

### 5.7.2.1 Modeling Player Experience in *Super Mario Bros*

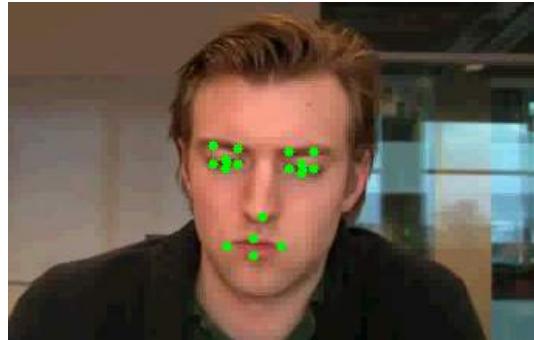
Our first example builds upon the work of Pedersen et al. [521, 520] who modified an open-source clone of the classic platform game *Super Mario Bros* (Nintendo, 1985) to allow for personalized level generation. That work is important in that it set the foundations for the development of the *experience-driven procedural content*

*generation* framework [783] which constitutes a core research trend within procedural content generation (see also Chapter 4).

The game used in this example is a modified version of Markus Persson's *Infinite Mario Bros* which is a public domain clone of Nintendo's classic platform game *Super Mario Bros* (Nintendo, 1985). All experiments reported in this example rely on a **model-free** approach for the modeling of player experience. Models of player experience are based on both the level played (game context) and the player's playing style. While playing, the game recorded a number of behavioral metrics of the players, such as the frequency of jumping, running and shooting, that are taken into consideration for modeling player experience. Further, in a follow-up experiment [610], the videos of players playing the game were also recorded and used to extract a number of useful visual cues such as the average head movement during play. The output (ground truth of experience) for all experiments is provided as first-person, rank-based reports obtained via **crowdsourcing**. Data was crowdsourced from hundreds of players, who played pairs of levels with different level parameters (e.g., dissimilar numbers, sizes and placements of gaps) and were asked to rank which of two levels best induced a number of player states. Across the several studies of player experience modeling for this variant of *Super Mario Bros* (Nintendo, 1985) collectively players have been asked to annotate fun, engagement, challenge, frustration, predictability, anxiety, and boredom. These are the target outputs the player model needs to predict based on the input parameters discussed above.

Given the rank-based nature of the annotations the use of preference learning is necessary for the construction of the player model. The collected data is used to train artificial neural networks that predict the players' experiential states, given a player's behavior (and/or affective manifestations) and a particular game context, using evolutionary preference learning. In **neuroevolutionary preference learning** [763], a genetic algorithm evolves an artificial neural network so that its output matches the pairwise preferences in the data set. The input of the artificial neural network is a set of features that have been extracted from the data set—as mentioned earlier, the input may include gameplay and/or objective data in this example. It is worth noting that automatic feature selection is applied to pick the set of features (model input) that are relevant for predicting variant aspects of player experience. The genetic algorithm implemented uses a fitness function that measures the difference between the reported preferences and the relative magnitude of the model output. Neuroevolutionary preference learning has been used broadly in the player modelling literature and the interested reader may refer to the following studies (among many): [432, 610, 763, 521, 520, 772].

The crowdsourcing experiment of Pedersen et al. [521, 520] resulted in data (gameplay and subjective reports of experience) from **181 players**. The best predictor of reported *fun* reached an accuracy of around 70% on unseen subjective reports of fun. The input of the neural network *fun*-model is obtained through automatic feature selection consists of the time Mario spent moving left, the number of opponents Mario killed from stomping, and the percentage of level played in the left direction. All three playing features appear to contribute positively for the prediction of reported *fun* in the game. The best-performing model for *challenge*



**Fig. 5.13** Facial feature tracking for head movement. Image adapted from [610].

prediction had an accuracy of approximately 78%. It is more complex than the best fun predictor, using five features: time Mario spends standing still, jump difficulty, coin blocks Mario pressed, number of cannonballs Mario killed, and Mario kills by stomping. Finally, the best predictor for *frustration* reaches an accuracy of 89%. It is indeed an impressive finding that a player experience model can predict (with near-certainty) whether the player is frustrated by the current game by merely calculating the time Mario spent standing still, the time Mario spent on its last life, the jump difficulty, and the deaths Mario had from falling in gaps. The general findings of Pedersen et al. [520] suggest that good predictors for experience can be found if a preference learning approach is applied on crowdsourced reports of experience and gameplay data. The prediction accuracies, however, depend on the complexity of the reported state—arguably *fun* is a much more complicated and fuzzier notion to report compared to *challenge* or *frustration*. In a follow up study by Pedersen et al. [521] the additional player states of *predictability*, *anxiety* and *boredom* were predicted with accuracies of approximately 77%, 70% and 61%, respectively. The same player experience methodology was tested on an even larger scale, soliciting data from a total number of **780 players** of the game [621]. Frequent pattern mining algorithms were applied to the data to derive frequent sequences of player actions. Using sequential features of gameplay the models reach accuracies of up to 84% for *engagement*, 83% for *frustration* and 79% for *challenge*.

In addition to the behavioral characteristics the **visual cues** of the player can be taken into account as objective input to the player model. In [610] visual features were extracted from videos of **58 players**, both throughout whole game sessions and during small periods of critical events such as when a player completes a level or when the player loses a life (see Figs. 5.13 and 5.14). The visual cues enhance the quality of the information we have about a player's affective state which, in turn, allows us to better approximate player experience. Specifically, fusing the gameplay and the visual reaction features as inputs to the artificial neural network we achieve average accuracies of up to 84%, 86% and 84% for predicting reported *engagement*, *frustration* and *challenge*, respectively. The key findings of [610] suggest



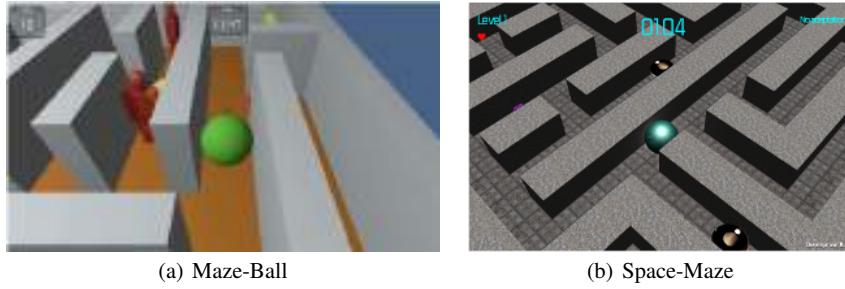
**Fig. 5.14** Examples of facial expressions of *Super Mario Bros* (Nintendo, 1985) players for different game states. All images are retrieved from the Platformer Experience Dataset [326].

that players' visual reactions can provide a rich source of information for modeling experience preferences and lead to more accurate models of player experience.

#### 5.7.2.2 Modeling Player Experience in *Maze-Ball*

Our second example for modeling player experience builds largely upon the extensive studies of Martínez et al. [434, 430, 435] who analyzed player experience using a simple 3D prey-predator game named *Maze-Ball* towards achieving affective-driven camera control in games. While the game is rather simple, the work on *Maze-Ball* offers a thorough analysis of player experience via a set of sophisticated techniques for capturing the psychophysiological patterns of players including preference learning, frequent pattern mining and deep convolutional neural networks. In addition the dataset that resulted from these studies is publicly available for further experimentation and forms a number of suggested exercises for this book.

*Maze-Ball* is a three-dimensional prey-predator game (see Fig. 5.15); similar to a 3D version of *Pac-Man* (Namco, 1981). The player (prey) controls a ball which



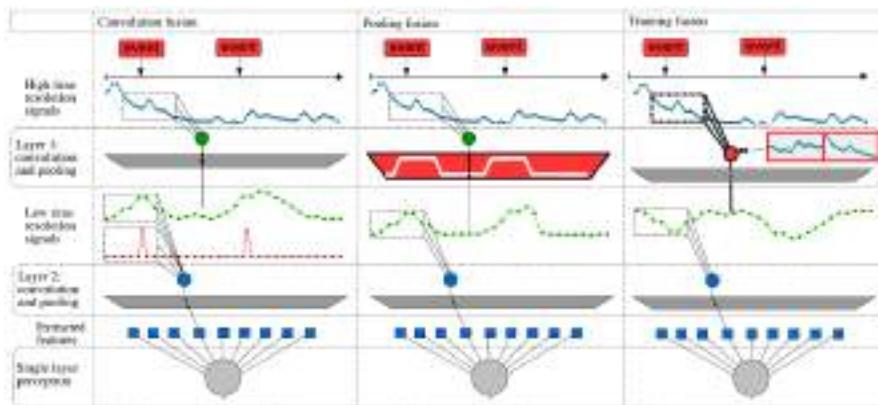
**Fig. 5.15** Early Maze-Ball prototype (a) and a polished variant of the game (b) that features real-time camera adaptation [345]. The games can be found and played at <http://www.hectorpmartinez.com/>.

moves inside a maze hunted by 10 opponents (predators) moving around the maze. The goal of the player is to maximize her score by gathering as many tokens, scattered in the maze, as possible while avoiding being touched by the opponents in a predefined time window of 90 seconds. A detailed description of *Maze-Ball* gameplay can be found in [780].

Gameplay attributes and physiological signals (skin conductance and heart rate variability) were acquired from **36 players** of *Maze-Ball*. Each subject played a pre-defined set of eight games for 90 seconds, thus the total number of game sessions available is 288. Gameplay and physiological signals define the **input** of the player experience model. To obtain the ground truth of experience players self-reported their preference about game pairs they played using a rank-based questionnaire, in particular a 4-alternative forced choice (4-AFC) protocol [780]. They were asked to rank the two games of each game pair with respect to *fun*, *challenge*, *boredom*, *frustration*, *excitement*, *anxiety* and *relaxation*. These annotations are the target **outputs** the player model will attempt to predict based on the input parameters discussed above.

Several features were extracted from the gameplay and physiological signals obtained. These included features related to kills and deaths in the game as well as features associated with the coverage of the level. For extracting features from the physiological signals the study considered their average, standard deviation, and maximum and minimum values. The complete feature list and the experimental protocol followed can be found in [780].

As in the example with the game variant of *Super Mario Bros* (Nintendo, 1985) the rank-based nature of the annotations requires the use of **preference learning** for the construction of the player experience model. Thus, the collected data is used to train neural networks that predict the player states, given a player's gameplay behavior and its physiological manifestations, using evolutionary preference learning. The architecture of the neural network can be either shallow using a simple multi-layered perceptron or deep (using convolutional neural networks [430, 435]). Figure 5.16 shows the different ways information from gameplay and physiology can be



**Fig. 5.16** Three dissimilar approaches to deep multimodal fusion via convolutional neural networks. Gameplay events are fused with skin conductance in this example. The networks illustrated present two layers with one neuron each. The first convolutional layer receives as input a continuous signal at a high time resolution, which is further reduced by a pooling layer. The resulting signal (feature map) presents a lower time resolution. The second convolutional layer can combine this feature map with additional modalities at the same low resolution. In the convolution fusion network (left figure), the two events are introduced at this level as a pulse signal. In the pooling fusion network (middle figure), the events are introduced as part of the first pooling layer, resulting in a filtered feature map. Finally, in the training fusion network (right figure), the events affect the training process of the first convolutional layer, leading to an alternative feature map. Image adapted from [435].

fused on a **deep convolutional neural network** which is trained via preference learning to predict player experience in any game experience dataset that contains discrete in-game events and continuous signals (e.g., the player's skin conductance).

Predictors of the player experience states can reach accuracies that vary from 72% for *challenge* up to 86% for *frustration* using a shallow multi-layer perceptron player model [780]. Significant improvements are observed in those accuracies when the input space of the model is augmented with frequent sequences of in-game and physiological events (i.e., fusion on the input space). As in [621], Martínez et al. used GSP to extract those frequent patterns that were subsequently used as inputs of the player model [434]. Further accuracy improvements can be observed when physiology is fused with physiological signals on deep architectures of convolutional neural networks [435]. Using **deep fusion** (see Fig. 5.16) accuracies of predicting player experience may surpass 82% for all player experience states considered. Further information about the results obtained in the *Maze-Ball* game can be found in the following studies: [780, 434, 430, 435, 436].

## 5.8 Further Reading

For an extensive reading on game and player analytics (including visualization, data preprocessing, data modeling and game domain-dependent tasks) we refer the reader to the edited book by El-Nasr et al. [186]. When it comes to player modeling two papers offer complementary perspectives and taxonomies of player modeling and a thorough discussion on what aspects of players can be modeled and the ways players can be modeled: the survey papers of Smith et al. [636] and Yannakakis et al. [782].

## 5.9 Exercises

In this section we propose a set of exercises for modeling both the behavior and the experience of game players. For that purpose, we outline a number of datasets that can be used directly for analysis. Please note, however, that the book's website will remain up to date with more datasets and corresponding exercises beyond the ones covered below.

### 5.9.1 Player Behavior

A proposed semester-long game data mining project is as follows. You have to choose a dataset containing player behavioral attributes to apply the necessary pre-processing on the data such as extracting features and selecting features. Then you must apply a relevant **unsupervised learning** technique for compressing, analyzing, or reducing the dimensionality of your dataset. Based on the outcome of unsupervised learning you will need to implement a number of appropriate **supervised learning** techniques that learn to predict a data attribute (or a set of attributes). We leave the selection of algorithms to the reader or the course instructor. Below we discuss a number of example datasets one might wish to start from; the reader, however, may refer to the book's website for more options on game data mining projects.

#### 5.9.1.1 SteamSpy Dataset

*SteamSpy* (<http://steamspy.com/>) is a rich dataset of thousands of games released on Steam<sup>6</sup> containing several attributes each. While strictly not a dataset focused on player modeling, *SteamSpy* offers an accessible and large dataset for game analytics. The data attributes of each game include the game's name, the developer, the publisher, the score rank of the game based on user reviews, the number of owners

---

<sup>6</sup> <http://store.steampowered.com/>

of the game on Steam, the people that have played this game since 2009, the people that have played this game in the last two weeks, the average and median playtime, the game's price and the game's tags. The reader may use an API<sup>7</sup> to download all data attributes from all games contained in the dataset. Then one might wish to apply supervised learning to be able to predict an attribute (e.g., the game's price) based on other game features such as the game's score, release date and tags. Or alternatively, one might wish to construct a score predictor of a new game. The selection of the modeling task and the AI methods is left to the reader.

### 5.9.1.2 *StarCraft: Brood War Repository*

The *StarCraft: Brood War* repository contains a number of datasets that include thousands of professional *StarCraft* replays. The various data mining papers, datasets as well as replay websites, crawlers, packages and analyzers have been compiled by Alberto Uriarte at Drexel University.<sup>8</sup> In this exercise you are faced with the challenge of mining game replays with the aim to predict a player's strategy. Some results on the *StarCraft: Brood War* datasets can be found in [750, 728, 570] among others.

## 5.9.2 *Player Experience*

As a semester project on player experience modeling it is suggested you choose a game, one or more affective or cognitive states to model (model's output) and one or more input modalities. You are expected to collect empirical data using your selected game and build models for the selected psychological state of the players that rely on the chosen input modalities.

As a smaller project that does not involve data collection you may opt to choose one of the following datasets and implement a number of AI methods that will derive accurate player experience models. The models should be compared in terms of a performance measure. The two datasets accompanying this book and outlined below are the **platformer experience dataset** and the **Maze-Ball dataset**. The book's website will be up to date with more datasets and exercises beyond the ones covered below.

### 5.9.2.1 Platformer Experience Dataset

The extensive analysis of player experience in *Super Mario Bros* (Nintendo, 1985) and our wish to further advance our knowledge and understanding on player expe-

---

<sup>7</sup> <http://steamspy.com/api.php>

<sup>8</sup> Available at: <http://nova.wolfwork.com/dataMining.html>

rience had led to the construction of the Platformer Experience Dataset [326]. This is the first open-access game experience corpus that contains multiple modalities of data from players of *Infinite Mario Bros*, a variant of *Super Mario Bros* (Nintendo, 1985). The open-access database can be used to capture aspects of player experience based on **behavioral** and **visual** recordings of platform game players. In addition, the database contains aspects of the **game context**—such as level attributes—demographic data of the players and self-reported annotations of experience in two forms: **ratings** and **ranks**.

Here are a number of questions you might wish to consider when attempting to build player experience models that are as accurate as possible: Which AI methods should I use? How should I treat my output values? Which feature extraction and selection mechanism should I consider? The detailed description of the dataset can be found here: <http://www.game.edu.mt/PED/>. The book website contains further details and a set of exercises based on this dataset.

#### 5.9.2.2 *Maze-Ball* Dataset

As in the case of the Platformer Experience Dataset the *Maze-Ball* dataset is also publicly available for further experimentation. This open-access game experience corpus contains two modalities of data obtained from *Maze-Ball* players: their **gameplay** attributes and three **physiological signals**: blood volume pulse, heart rate and skin conductance. In addition, the database contains aspects of the game such as features of the virtual camera placement. Finally the dataset contains demographic data of the players and self-reported annotations of experience in two forms: **ratings** and **ranks**.

The aim, once again, is to construct the most accurate models of experience for the players of *Maze-Ball*. So, which modalities of input will you consider? Which annotations are more reliable for predicting player experience? How will your signals be processed? These are only a few of the possible questions you will encounter during your efforts. The detailed description of the dataset can be found here: <http://www.hectorpmartinez.com/>. The book website contains further details about this dataset.

## 5.10 Summary

This chapter focused on the use of AI for modeling players. The core reasons why AI should be used for that purpose is either to derive something about the players' experience (how they feel in a game) or for us to understand something about their behavior (what they do in a game). In general we can model player behavior and player experience by following a top-down or a bottom-up approach (or a mix of the two). Top-down (or model-based) approaches have the advantage of solid theoretical frameworks usually derived from other disciplines or other domains than

games. Bottom-up (or model-free) instead rely on data from players and have the advantage of not assuming anything about players other than that player experience and behavior are associated with data traces left by the player and that these data traces are representative of the phenomenon we wish to explain. While a hybrid between model-based and model-free approaches is in many ways a desirable approach to player modeling, we focus on bottom-up approaches, where we provide a detailed taxonomy for the options available regarding the input and the output of the model, and the modeling mechanism per se. The chapter ends with a number of player modeling examples, for modeling both the behavior of players and their experience.

The player modeling chapter is the last chapter of the second part of this book, which covered the core uses of AI in games. The next chapter introduces the third and last part of the book, which focuses on the holistic synthesis of the various AI areas, the various methods and the various users of games under a common game AI framework.



## **Part III**

### **The Road Ahead**



## Chapter 6

# Game AI Panorama

This chapter attempts to give a high-level overview of the field of game AI, with particular reference to how the different core research areas within this field inform and interact with each other, both actually and potentially. For that purpose we first identify the main research areas and their sub-areas within the game AI field. We then view and analyze the areas from three key perspectives: (1) the dominant AI method(s) used under each area; (2) the relation of each area with respect to the end (human) user; and (3) the placement of each area within a human-computer (player-game) interaction perspective. In addition, for each of these areas we consider how it could inform or interact with each of the other areas; in those cases where we find that meaningful interaction either exists or is possible, we describe the character of that interaction and provide references to published studies, if any.

The main motivations for us writing this chapter is to help the reader understand how a particular area relates to other areas within this increasingly growing field, how the reader can benefit from knowledge created in other areas and how the reader can make her own research more relevant to other areas. To facilitate and foster synergies across active research areas we place all key studies into a taxonomy with the hope of developing a common understanding and vocabulary within the field of AI and games. The structure of this chapter is based on the first holistic overview of the game AI field presented in [785]. The book takes a new perspective on the key game AI areas given its educational and research focus.

The main game AI areas and core subareas already identified in this book and covered in this chapter are as follows:

- **Play Games** (see Chapter 3) which includes the subareas of *Playing to Win* and *Playing for the Experience*. Independently of the purpose (winning or experience) AI can control either the player character or the non-player character.
- **Generate Content** (see Chapter 4) which includes the subareas of *autonomous (procedural) content generation* and *assisted content generation*. Please note that the terms *assisted (procedural) content generation* and *mixed-initiative (procedural) content generation* (as defined in Chapter 4) are used interchangeably in this chapter.

- **Model Players** (see Chapter 5) which includes the subareas of *player experience modeling* and *player behavior modeling*, or else, *game data mining* [178].

The scope of this chapter is not to provide an inclusive survey of all game AI areas—the details of each area have been covered in preceding chapters of the book—but rather a roadmap of interconnections between them via representative examples. As research progresses in this field, new research questions will pop up and new methods be invented, and other questions and methods recede in importance. We believe that all taxonomies of research fields are by necessity tentative. Consequently, the list of areas defined in this chapter should not be regarded as fixed and final.

The structure of the chapter is as follows: In Section 6.1, we start by holistically analyzing the game AI areas within the game AI field and we provide three alternative views over game AI: one with respect to the methods used, one with respect to the end users within game research and development and one where we outline how each of the research areas fits within the player-game interaction loop of digital games. Then, Section 6.2, digs deeper into the research areas and describes each one of them in detail. With the subsection describing each area, there is a short description of the area and a paragraph on the possible interactions with each of the other areas for which we have been able to identify strong or weak influences . The chapter ends with a section containing our key conclusions and vision for the future of the field.

## 6.1 Panoramic Views of Game AI

Analyzing any research field as a composition of various subareas with interconnections and interdependencies can be achieved in several different ways. In this section we view game AI research from three high-level perspectives that focus on the computer (i.e., the AI methods), the human (i.e., the potential end user of game AI) and the interaction between the key end user (i.e., player) and the game. Instead in Section 6.2 we outline each game AI area and present the interconnections between the areas.

Game AI is composed of (a set of) methods, processes and algorithms in artificial intelligence as those are applied to, or inform the development of, games. Naturally, game AI can be analyzed through the **method** used by identifying the dominant AI approaches under each game AI area (see Section 6.1.1). Alternatively, game AI can be viewed from the game domain perspective with a focus on the **end users** of each game AI area (see Section 6.1.2). Finally game AI is, by nature, realized through systems that entail rich human-computer interaction (i.e., games) and, thus, the different areas can be mapped to the interaction framework between the player and the game (see Section 6.1.3).

**Table 6.1** Dominant (●) and secondary (○) AI methods for each of the core AI areas we cover in this book. The total number of methods used for each area appears at the bottom row of the table.

|                          | Play Games<br>Winning | Generate Content<br>Autonomously | Model Players<br>Experience |       |       |
|--------------------------|-----------------------|----------------------------------|-----------------------------|-------|-------|
|                          | Experience            | Assisted                         | Behavior                    |       |       |
| Behavior Authoring       | ●                     | ●                                |                             |       |       |
| Tree Search              | ●                     | ○                                | ○                           | ○     |       |
| Evolutionary Computation | ●                     | ○                                | ●                           | ●     | ●     |
| Supervised Learning      | ○                     | ●                                |                             | ●     | ●     |
| Reinforcement Learning   | ●                     | ○                                |                             |       |       |
| Unsupervised Learning    |                       |                                  |                             | ○     | ○     |
| Total (Dominant)         | 5 (4)                 | 5 (2)                            | 2 (1)                       | 3 (1) | 3 (2) |
|                          |                       |                                  |                             |       | 2 (2) |

### 6.1.1 Methods (Computer) Perspective

The first panoramic view of game AI we present is centered around the AI methods used in the field. As the basis of this analysis we first list the core AI methods mostly used in the game AI field. The key methodology areas identified in Chapter 2 include ad-hoc behavior authoring, tree search, evolutionary computation, reinforcement learning, supervised learning, and unsupervised learning. For each of the game AI areas investigated we have identified the AI methods that are **dominant** or **secondary** in the area. While the dominant methods represent the most popular techniques used in the literature, secondary methods represent techniques that have been considered from a substantial volume of studies but are not dominant.

We have chosen to group methods according to what we perceive as a received taxonomy and following the structure of Chapter 2. While it would certainly be possible to classify the various methods differently, we argue that the proposed classification is compact (containing solely key methodology areas) and it follows standard method classifications in AI. While this taxonomy is commonly accepted, the lines can be blurred. In particular evolutionary computation, being a very general optimization method, can be used to perform supervised, unsupervised or reinforcement learning (more or less proficiently). The model-building aspect of reinforcement learning can be seen as a supervised learning problem (mapping from action sequences to rewards), and the commonly used tree search method Monte Carlo tree search can be seen as a form of TD learning. The result of any tree search algorithm can be seen as a plan, though it is often not guaranteed to lead to the desired end state. That the various methods have important commonalities and some overlap does not detract from the fact that each of them is clearly defined.

Table 6.1 illustrates the relationship between game AI areas and corresponding methods. It is evident that evolutionary computation and supervised learning appear to be of dominant or secondary use in most game AI areas. Evolutionary computation is a dominant method for playing to win, for generating content (in an assisted/mixed-initiative fashion or autonomously), and for modeling players; it has also been considered for the design of believable play (play for experience) research. Supervised learning is of substantial use across the game AI areas and appears to be

dominant in player experience and behavioral modeling, as well as in the area of AI that plays for experience. Behavior authoring, on the other hand, is useful solely for game-playing. Reinforcement learning and unsupervised learning find limited use across the game AI areas, respectively, being dominant only on AI that plays to win and player behavior modeling. Finally, tree search finds use primarily in playing to win and it is also considered—as a form of planning—for controlling play for experience and in computational narrative (as part of autonomous or assisted PCG).

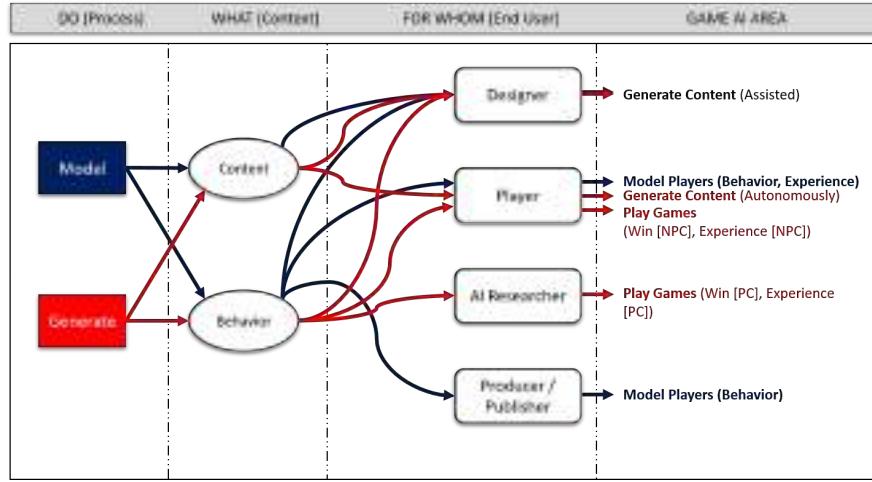
Viewing Table 6.1 from the game AI areas' perspective (columns) it seems that AI that plays games (either for winning or for the experience) defines the game AI area with the most diverse and richest palette of AI methods. On the contrary, procedural content generation is solely dominated by evolutionary computation and tree search to a secondary degree. It is important to state that the popularity of any AI method within a particular area is closely tied to the task performed or the goal in mind. For example, evolutionary computation is largely regarded as a computationally heavy process which is mostly used in tasks associated with offline training. As PCG so far mainly relies on content that is generated offline, evolutionary computation offers a good candidate method and the core approach behind search-based PCG [720]. If online learning is a requirement for the task at hand, however, other methods (such as reinforcement learning or pruned tree-search) tend to be preferred.

Clearly the possibility space for future implementations of AI methods under particular game AI areas seems rather large. While particular methods have been traditionally dominant in specific areas for good reasons (e.g., planning in computational narrative) there are equally good reasons to believe that the research in a game AI area itself has been heavily influenced by (and limited to) its corresponding dominant AI methods. The empty cells of Table 6.1 indicate potential areas for exploration and offer us an alternative view of promising new intersections between game AI areas and methods.

### 6.1.2 End User (Human) Perspective

The second panoramic view of the game AI field puts an emphasis on the end user of the AI technology or general outcome (product or solution). Towards that aim we investigate three core dimensions of the game AI field and classify all game AI areas with respect to the **process** AI follows, the **game context** under which algorithms operate and, finally, the **end user** that benefits most from the resulting outcome. The classes identified under the above dimensions are used as the basis of the taxonomy we propose.

The first dimension (phrased as a question) refers to the AI process: *In general, what can AI do within games?* We identify two potential classes in this dimension: AI can **model** or **generate**. For instance, an artificial neural network can model a playing pattern, or a genetic algorithm can generate game assets. Given that AI can model or generate the second dimension refers to the context: *What can AI methods model or generate in a game?* The two possible classes here are **content**



**Fig. 6.1** The end user perspective of the identified game AI areas. Each AI area follows a **process** (model or generate) under a **context** (content or behavior) for a particular **end user** (designer, player, AI researcher or game producer/publisher). Blue and red arrows represent the processes of modeling and generation, respectively. Modified graph from [785].

and **behavior**. For example, AI can model a players' affective state, or generate a level. Finally, the third dimension is the end user: *AI can model, or generate, either content or behavior; but, for whom?* The classes under the third dimension are the **designer**, the **player**, the **AI researcher**, and the **producer/publisher**.

Note that the above taxonomy serves as a framework for classifying the game AI areas according to the end user and is, by no means, inclusive of all potential processes, contexts, and end users. For instance, one could claim that the producer's role should be distinct from the publisher's role and that a developer should also be included in that class. Moreover, game content could be further split into smaller sub-classes such as narrative, levels, etc. Nevertheless, the proposed taxonomy provides distinct roles for the AI process (model vs. generate vs. evaluate), clear-cut classification for the context (content vs. behavior) and a high-level classification of the available stakeholders in game research and development (designer vs. player vs. AI researcher vs. producer/publisher). The taxonomy presented here is a modified version of the one introduced in [785] and it does not consider **evaluation** as a process for AI since it is out of the primary scope of this book.

Figure 6.1 depicts the relationship between the game AI core areas, the subareas and the end users in game research and development. Assisted, or mixed-initiative, content generation is useful for the designer and entails all possible combinations of processes and context as both content and behavior can be either modeled or generated for the designer. Compared to the other stakeholders the player benefits directly from more game AI research areas. In particular the player and her experience are affected by research on player modeling, which results from the modeling of experience and behavior; research on autonomous procedural content generation, as

a result of generation of content; and studies on NPC playing (for winning or experience) resulting from the generation of behavior. The player character (PC)-based game playing (for winning or experience) areas provide input to the AI researcher primarily. Finally, the game producer/publisher is primarily affected by results on behavioral player modeling, game analytics and game data mining as a result of behavior modeling.

### 6.1.3 Player-Game Interaction Perspective

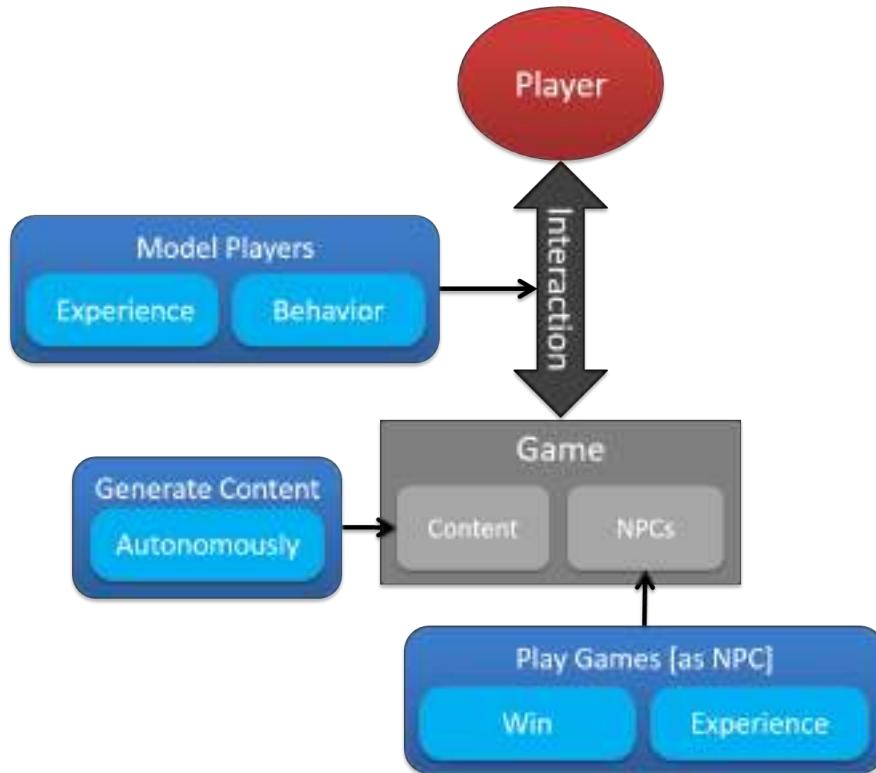
The third and final panoramic perspective of game AI presented in this section couples the computational processes with the end user within a game and views all game AI areas through a human-computer interaction—or, more accurately, a player-game interaction—lens. The analysis builds on the findings of Section 6.1.2 and places the five game AI areas that concern the player as an end user on a player-game interaction framework as depicted in Fig. 6.2. Putting an emphasis on player experience and behavior, player modeling directly focuses on the interaction between a player and the game context. Game content is influenced primarily by research on autonomous procedural content generation. In addition to other types of content, most games feature NPCs, the behavior of which is controlled by some form of AI. NPC behavior is informed by research in NPCs that play the game to win or any other playing-experience purpose such as believability.

Looking at the player-game interaction perspective of game AI it is obvious that the player modeling area has the most immediate and direct impact on the player experience as it is the only area linked to the player-game interaction directly. From the remaining areas, PCG influences player experience the most as all games have some form of environment representation and mechanics. Finally, AI that plays as an NPC (either to win or for the experience of play) is constrained to games that include agents or non-player characters.

The areas not considered directly in this game AI perspective affect the player rather remotely. Research on AI tools that assist the generative process of content improves the game’s quality as a whole and in retrospect the player experience since designers tend to maintain a *second-order player model* [378] while designing. Finally, AI that plays the game as a player character can be offered for testing both the content and the NPC behaviors of a game, but also the interaction between the player and the game (via e.g., player experience competitions), but is mainly directed to AI researchers (see Fig. 6.1).

## 6.2 How Game AI Areas Inform Each Other

In this section, we outline the core game AI areas and discuss how they inform or influence (the terms are used interchangeably) each other. All research areas could



**Fig. 6.2** Game AI areas and sub-areas viewed from a player-game interaction perspective.

be seen as potentially influencing each other to some degree; however, making a list of all such influences would be impractical and the result would be uninteresting. Therefore we only describe *direct* influences. Direct influences can be either **strong** (represented by a • as the bullet point style next to the corresponding influence in the following lists) or **weak** (represented by a ○). We do not list influences we do not consider potentially important for the informed research area, or which only go through a third research area.

The sections below list **outgoing** influence. Therefore, to know how area *A* influences area *B* you should look in the section describing area *A*. Some influences are mutual, some not. The notation  $A \rightarrow B$  in the headings of this section denotes that “*A* influences *B*”. In addition to the text description each section provides a figure representing all outgoing influences of the area as arrows. Dark and light gray colored areas represent, respectively, *strong* and *weak* influence. Areas with white background are not influenced by the area under consideration. The figures also depict the incoming influences from other areas. Incoming *strong* and *weak* influences are represented, respectively, with a solid line and a dashed line around the game AI areas that influence the area under consideration. Note that the description of the

incoming influence from an area is presented in the corresponding section of that area.

### **6.2.1 Play Games**

The key area in which AI plays a game (as covered in Chapter 3) involves the sub-areas of **Playing to Win** and **Playing for Experience**. As mentioned earlier in the chapter the AI can control either player or non-player characters of the game. We cover the influences to (and from) these subareas of game AI in this section.

#### **6.2.1.1 Playing to Win (as a Player or as a Non-Player)**

As already seen in Chapter 3 research in AI that learns to play (and win) a game focuses on using **reinforcement learning** techniques such as temporal difference learning or evolutionary algorithms to learn policies/behaviors that play games well—whether it is a PC or an NPC playing the game. From the very beginning of AI research, reinforcement learning techniques have been applied to learn how to play board games (see for example Samuel’s Checkers player [591]). Basically, playing the game is seen as a reinforcement learning problem, with the reinforcement tied to some measure of success in the game (e.g., the score, or length of time survived). As with all reinforcement learning problems, different methods can be used to solve the problem (find a good policy) [715] including TD learning [689], evolutionary computation [406], competitive co-evolution [24, 538, 589, 580], simulated annealing [42], other optimization algorithms and a large number of combinations between such algorithms [339]. In recent years a large number of papers that describe the application of various learning methods to different types of video games have appeared in the literature (including several overviews [470, 406, 632, 457]). Finally, using games to develop artificial general intelligence builds on the idea that games can be useful environments for algorithms to learn complex and useful behaviors; thus research in algorithms that learn to win is essential.

It is also worth noting that most existing game-based **benchmarks** measure how well an agent plays a game—see for example [322, 404, 504]. Methods for learning to play a game are vital for such benchmarks, as the benchmarks are only meaningful in the context of the algorithms. When algorithms are developed that “beat” existing benchmarks, new benchmarks need to be developed. For example, the success of an early planning agent in the first Mario AI competition necessitated that the software be augmented with a better level generator for the next competition [322], and for the Simulated Car Racing competition, the performance of the best agents on the original competition game spurred the change to a new more sophisticated racing game [710, 392].



**Fig. 6.3** Playing to Win: influence on (and from) other game AI research areas. **Outgoing** influence (represented by arrows): black and dark gray colored areas reached by arrows represent, respectively, **strong** and **weak** influence. **Incoming** influence is represented by red lines around the areas that influence the area under investigation (i.e., AI that plays to win in this figure): **strong** and **weak** influences are represented, respectively, by a solid and a dashed line.

Research in this area impacts game AI at large as three game AI subareas are directly affected; in turn, one subarea is directly affecting AI that plays to win (see Fig. 6.3).

- **Playing to Win → Playing for the Experience:** An agent cannot be believable or existent to augment the game's experience if it is not proficient. Being able to play a game well is in several ways a precondition for playing games in a believable manner though well playing agents can be developed without learning (e.g., via top-down approaches). In recent years, successful entries to competitions focused on believable agents, such as the 2K BotPrize and the Mario AI Championship Turing test track, have included a healthy dose of learning algorithms [719, 603].
- **Playing to Win → Generate Content (Autonomously):** Having an agent that is capable of playing a game proficiently is useful for **simulation-based testing** in procedural content generation, i.e., the testing of newly generated game content by playing through that content with an agent. For example, in a program generating levels for the platform game *Super Mario Bros* (Nintendo, 1985), the levels can be tested by allowing a trained agent to play them; those that the agent cannot complete can be discarded [335]. Browne's *Ludi* system, which generates complete board games, evaluates these games through simulated playthrough and uses learning algorithms to adapt the strategy to each game [74].
- **Playing to Win → Generate Content (Assisted):** Just as with autonomous procedural content generation, many tools for AI-assisted game design rely on being able to simulate playthroughs of some aspect of the game. For instance, the *Sentient Sketchbook* tool for level design uses simple simulations of game-playing agents to evaluate aspects of levels as they are being edited by a human de-

signer [379]. Another example is the automated playtesting framework named *Restricted Play* [295] which aims mostly at assisting designers on aspects of game balance during game design. A form of *Restricted Play* is featured in the *Ludocore* game engine [639].

### 6.2.1.2 Playing for the Experience (as a Player or as a Non-Player)

Research on AI that plays a game for a purpose other than winning is central to studies where playing the game well is not the primary research aim. AI can play the game as a *player* character attempting to maximize the believability value of play as, for instance, in [719, 619, 96]. It can alternatively play the game in a role of an NPC for the same purpose [268]. Work under this research subarea involves the study of believability, interestingness or playing experience in games and the investigations of mechanisms for the construction of **agent architectures** that appear to have e.g., believable or human-like characteristics. The approaches for developing such architectures can be either top-down behavior authoring (such as the FAtiMA model used in *My Dream Theatre* [100] and the Mind Module model [191] used in *The Pataphysic Institute*) or bottom-up attempting to imitate believable gameplay from human players such as the early work of Thurau et al. in *Quake II* (Activision, 1997) bots [696], the human imitation attempts in *Super Mario Bros* (Nintendo, 1985) [511], the *Unreal Tournament 2004* (Epic Games, 2004) believable bots of Schrum et al. [603] and the crowdsourcing studies of the *Restaurant game* [508]. Evidently, commercial games have long benefited from agent believability research. Examples of this include popular games such as the *Sims* (Electronic Arts, 2000) series. The industry puts a strong emphasis on the design of believability in games as this contributes to more immersive game environments. The funding of believability research through game AI competitions such as the 2K BotPrize is one of the many clear indicators of the commercial value of agent believability.

Over the last few years there has been a growing academic (and commercial) interest in the establishment of **competitions** that can be used as assessment tools for agent believability [719]. Agent believability research has provided input and given substance to those game benchmarks. A number of game Turing competitions have been introduced to the benefit of agent believability research, including the 2K BotPrize on the *Unreal Tournament 2004* (Epic Games, 2004) [647, 264] game and the Mario AI Championship: Turing test track [619] on the *Super Mario Bros* (Nintendo, 1985) game. Recently, the community saw AI agents passing the Turing test in the 2K BotPrize [603].

The study of AI that plays games not for winning, but for other purposes, affects research on three other game AI areas as illustrated in Fig. 6.4, whereas it is affected by four other game AI areas.

- **Playing for the Experience → Model Players (Experience and Behavior):**  
There is a direct link between player modeling and believable agents as research carried out for the modeling of human, human-like, and supposedly believable playing behavior can inform the construction of more appropriate models for



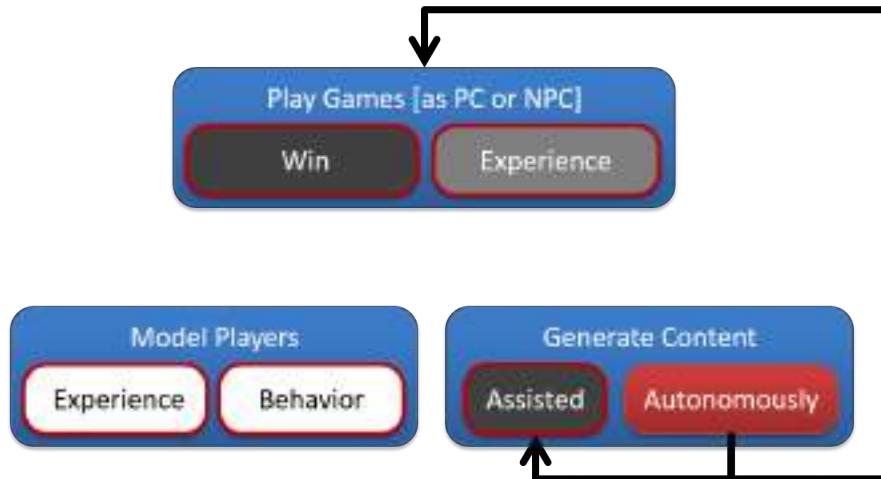
**Fig. 6.4** Playing for the Experience: influence on (and from) other game AI research areas.

players. Examples include the imitation of human play styles in *Super Mario Bros* (Nintendo, 1985) [511] and *Quake II* (Activision, 1997) [696]. Though computational player modeling uses learning algorithms, it is only in some cases that it is the behavior of an NPC that is modeled. In particular, this is true when the in-game behavior of one or several players is modeled. This can be done using either reinforcement learning techniques, or supervised learning techniques such as backpropagation or decision trees. In either case, the intended outcome for the learning algorithm is not necessarily an NPC that plays as well as possible, but one that plays in the style of the modeled player [735, 511].

- **Playing for the Experience → Generate Content (Autonomously):** Believable characters may contribute to better levels [96], more believable stories [801, 401, 531] and, generally, better game representations [563]. A typical example of the integration of characters in the narrative and the drive of the latter based on the former includes the FAtiMa agents in *FearNot!* [516] and *My Dream Theater* [100]. Another example is the generation of *Super Mario Bros* (Nintendo, 1985) levels that maximize the believability of any Mario player [96].

### 6.2.2 Generate Content

As covered in detail in Chapter 4 AI can be used to design whole (or parts of) games in an autonomous or in an assisted fashion. This core game AI area includes the subareas of **autonomous (procedural) content generation** and **assisted or mixed-initiative (procedural) content generation**. The interactions of these subareas with the remaining areas of game AI are covered in this section.



**Fig. 6.5** Generate Content (Autonomously): influence on (and from) other game AI research areas.

#### 6.2.2.1 Generate Content (Autonomously)

As stated in Chapter 4 procedural content generation has been included in limited roles in some commercial games since the early 1980s; however, recent years have seen an expansion of **research** on more controllable PCG for multiple types of game content [764], using techniques such as evolutionary search [720] and constraint solving [638]. The influence of PCG research beyond games is already evident in areas such as computational creativity [381] and interaction design (among others). There are several surveys of PCG available, including a recent book [616] and vision paper [702], as well as surveys of frameworks [783], sub-areas of PCG [554, 732] and methods [720, 638].

Autonomous content generation is one of the areas of recent academic research on AI in games which bears most promise for incorporation into **commercial games**. A number of recent games have been based heavily on PCG, including independent (“indie”) game production successes such as *Spelunky* (Mossmouth, 2009) and *Minecraft* (Mojang, 2011), and mainstream AAA games such as *Diablo III* (Blizzard Entertainment, 2012) and *Civilization V* (2K Games, 2010). A notable example, as mentioned in Chapter 4 is *No Man’s Sky* (Hello Games, 2016) with its quintillion different procedurally generated planets. Some games heavily based on PCG and developed by researchers have been released as commercial games on platforms such as Steam and Facebook; two good examples of this are *Petalz* [565, 566] and *Galactic Arms Race* [249].

Figure 6.5 depicts the three (and five) areas that are influenced by (and influence) autonomous PCG.

- **Generate Content (Autonomously) → Play to Win:** If an agent is trained to perform well in only a single game environment, it is easy to overspecialize the

training and arrive at a policy/behavior that will not generalize to other levels. Therefore, it is important to have a large number of environments available for training. PCG can help with this, potentially providing an infinite supply of test environments. For example, when training players for the Mario AI Championship it is common practice to test each agent on a large set of freshly generated levels, to avoid overtraining [322]. There has also been research on adapting NPC behavior specifically to generated content [332]. Finally, one approach to **artificial general intelligence** is to train agents to be good at playing games in general, and test them on a large variety of games drawn from some genre or distribution. To avoid overfitting, this requires games to be generated automatically, a form of PCG [598]. The generation of new environments is very important for NPC behavior learning, and this extends to benchmarks that measure some aspect of NPC behavior. Apart from the Mario AI Championship, competitions such as the Simulated Car Racing Championship use freshly generated tracks, unseen by the participants, to test submitted controllers [102]. But there is also scope for benchmarks and competitions focused on measuring the capabilities of PCG systems themselves, such as the Level Generation track of the Mario AI Championship [620].

- **Generate Content (Autonomously) → Play for the Experience:** Research on autonomous PCG naturally influences research on agent (PC or NPC) control for believability, interestingness or other aims aside from winning given that these agents are performing in a particular environment and under a specific game context. This influence is still in its infancy and the only study we can point the reader to is the one by Camilleri et al. [96] where the impact of level design on player character believability is examined in *Super Mario Bros* (Nintendo, 1985). Further, research on interactive narrative benefits from and influences the use of believable agents that interact with the player and are interwoven in the story plot. The narrative can yield more (or less) believability to agents and thus the relationship between the behavior of the agents and the emergent story is strong [801, 401, 531]. In that sense, the computational narrative of a game may define the arena for believable agent design.
- **Generate Content (Autonomously) → Generate Content (Assisted):** As content design is a central part of game design, many AI-assisted design tools incorporate some form of assisted content design. Examples include *Tanagra*, which helps designers create complete platform game levels which ensure playability through the use of constraint solvers [641], and *SketchaWorld* [634]. Another example is *Sentient Sketchbook*, which assists humans in designing strategy game levels through giving immediate feedback on properties of levels and autonomously suggesting modifications [379].

### 6.2.2.2 Generate Content (Assisted)

Assisted content generation refers to the development of AI-powered tools that support the game design and development process. This is perhaps the AI research area



**Fig. 6.6** Generate Content (Assisted): influence on (and from) other game AI research areas.

which is most promising for the development of better games [764]. In particular, AI can assist in the creation of game content varying from levels and maps to game mechanics and narratives. The impact of AI-enabled authoring tools on design and development influences the study of AI that plays games for believability, interestingness or player experience, and research in autonomous procedural content generation (see Fig. 6.6). **AI-assisted game design** tools range from those designed to assist with generation of complete game rulesets such as *MetaGame* [522] or *RuLearn* [699] to those focused on more specific domains such as strategy game levels [379], platform game levels [642], horror games [394] or physics-based puzzles [613].

It is worth noting that AI tools have been used extensively for supporting design and **commercial game development**. Examples such as the *SpeedTree* (Interactive Data Visualization Inc., 2013) generator for trees and other plants [287] have seen uses in several game productions. The mixed-initiative PCG tools mentioned above have a great potential in the near future as most of these are already tested on commercial games or developed with game industrial partners. Furthermore, there are tools designed for interactive modeling and analysis of game rules and mechanics, which are not focused on generating complete games but on prototyping and understanding aspects of complex games; such systems could be applied to existing commercial games [639].

- **Generate Content (Assisted) → Play for the Experience:** Authoring tools in forms of open-world sandboxes could potentially be used for the creation of more believable behaviors. While this is largely still an unexplored area of research and development, notable attempts include the NERO game AI platform where players can train game agents for efficient and believable first-person shooter bot behaviors [654]. An open version of this platform focused on crowdsourcing behaviors has been released recently [327]. A similar line of research is the gen-

eration of *Super Mario Bros* (Nintendo, 1985) players by means of interactive evolution [648].

- **Generate Content (Assisted) → Generate Content (Autonomously):** Research on methods of **mixed-initiative co-creation** [774] and design can feed input to and spur discussion on central topics in procedural content generation. Given the importance of content design in the development process as a whole, any form of mixed-initiative AI assistance in the generation process can support and augment procedural content generation. Notable examples of mixed-initiative PCG include the *Tanagra* platform game level design AI assistant [641], and the *SketchaWorld* [634], the *Sentient World* [380], the *Sentient Sketchbook* [379, 774] and the *Sonancia* [394] systems which generate game maps and worlds in a mixed-initiative design fashion following different approaches and levels of human computation. Further, tools can assist the authoring of narrative in games. In particular, **drama management** tools have long been investigated within the game AI community. An academic example is ABL which has allowed the authoring of narrative in *Façade* [441]. Among the few available and well-functional authoring tools the most notable is the *Versu* [197] storytelling system which was used in the game *Blood & Laurels* (Emily Short, 2014) and the *Inform 7* [480] software package that led to the design of *Mystery House Possessed* (Emily Short, 2005). More story generation tools as such can be found at the <http://storygen.org/> repository, by Chris Martens and Rogelio E. Cardona-Rivera.

### 6.2.3 Model Players

As already explored in Chapter 5, modeling players involves the subtasks of modeling their **behavior** or their **experience**. Given the interwoven nature of these two tasks we present their influences to (and from) other game AI areas under one common section. In player modeling [782, 636], computational models are created for detecting how the player perceives and reacts to gameplay. As stated in Chapter 5 such models are often built using machine learning methods where data consisting of some aspect of the game or player-game interaction is associated with labels derived from some assessment of player *experience*, gathered for example from questionnaires [781]. However, the area of player modeling is also concerned with structuring observed player *behavior* even when no correlates to experience are available—e.g., for identifying player types or predicting player behavior.

Research and development in player modeling can inform attempts for player experience in **commercial-standard games**. Player experience detection methods and algorithms can advance the study of user experience in commercial games. In addition, the appropriateness of sensor technology, the technical plausibility of biofeedback sensors, and the suitability of various modalities of human input can inform industrial developments. Quantitative testing via game metrics—varying from behavioral data mining to in-depth low scale studies—is also improved [764, 178, 186].



**Fig. 6.7** Model Players: influence on (and from) other game AI research areas.

By now, a considerable number of academic studies use directly datasets from commercial games to induce models of players that could inform further development of the game. For example, we refer the reader to the experiments in clustering players of *Tomb Raider: Underworld* (Square Enix, 2008) into archetypes [176] and predicting their late-game performance based on early-game behavior [414]. Examples of player modeling components within high-profile commercial games include the arousal-driven appearance of NPCs in *Left 4 Dead 2* (Valve Corporation, 2009), the fearful combat skills of the opponent NPCs in *F.E.A.R.* (Monolith, 2005), and the avatars' emotion expression in the *Sims* series (Maxis, 2000) and *Black and White* (Lionhead Studios, 2001). A notable example of a game that is based on player experience modeling is *Nevermind* (Flying Mollusk, 2016); the game adapts its content based on the stress of the player, which is manifested via a number of physiological sensors.

Player modeling is considered to be one of the core non-traditional uses of AI in games [764] and affects research in AI-assisted game design, believable agents, computational narrative and procedural content generation (see Fig. 6.7).

- **Model Players → Play for the Experience:** Player models can inform and update believable agent architectures. Models of behavioral, affective and cognitive aspects of gameplay can improve the human-likeness and believability of any agent controller—whether it is ad-hoc designed or built on data derived from gameplay. While the link between player modeling and believable agent design is obvious and direct, research efforts towards this integration within games are still sparse. However, the few efforts made on the imitation of human game playing for the construction of believable architectures have resulted in successful outcomes. For example, human behavior imitation in platform [511] and racing games [735, 307] has provided human-like and believable agents while similar approaches for developing *Unreal Tournament 2004* (Epic Games, 2004) bots (e.g., in [328]) recently managed to pass the Turing test in the 2K BotPrize com-

petition. Notably, one of the two agents that passed the Turing test in 2K BotPrize managed to do so by imitating (mirroring) aspects of human play [535]. A line of work that stands in between player modeling and playing games for the experience is the study on *procedural personas* [268, 267, 269]. As introduced in Chapter 5 procedural personas are NPCs that are trained to imitate realistically the decision making process of humans during play. Their study both influences our understanding about the internal (cognitive) processes of playing behavior and advances our knowledge on how to build believable characters in games.

- **Model Players → Generate Content (Autonomously):** There is an obvious link between computational models of players and PCG as player models can drive the generation of new personalized content for the player. The **experience-driven** role of PCG [783], as covered in Chapter 4, views game content as an indirect building block of a player’s affective, cognitive and behavioral state and proposes adaptive mechanisms for synthesizing personalized game experiences. The “core loop” of an experience-driven PCG solution involves learning a model that can predict player experience, and then using this model as part of an evaluation function for evolving (or otherwise optimizing) game content; game content is evaluated based on how well it elicits a particular player experience, according to the model. Examples of PCG that are driven by player models include the generation of game rules [716], camera profiles [780, 85] and platform game levels [617]. Most work that goes under the label “game adaptation” can be said to implement the experience-driven architecture; this includes work on adapting the game content to the player using reinforcement learning [28] or semantic constraint solving [398] rather than evolution. Player models may also inform the generation of computational narrative. Predictive models of playing experience can drive the generation of individualized scenarios in a game. Examples of the coupling between player modeling and computational narrative include the affect-driven narrative systems met in *Façade* [441] and *FearNot!* [26], and the affect-centered game narratives such as the one of *Final Fantasy VII* (Square, 1997).
- **Model Players → Generate Content (Assisted):** User models can enhance authoring tools that, in turn, can assist the design process. The research area that bridges user modeling and AI-assisted design is in its infancy and only a few example studies can be identified. Indicatively, **designer models** [378] have been employed to personalize mixed-initiative design processes [774, 377, 379]. Such models may drive the procedural generation of designer-tailored content.

### 6.3 The Road Ahead

This chapter has initially identified the currently most active areas and subareas within game AI and placed them on three holistic frameworks: an AI method mapping, a game stakeholder (end user) taxonomy and the player-game interaction loop. This analysis revealed dominant AI algorithms within particular areas as well as

room for exploration of new methods within areas. In addition, it revealed the dissimilar impact of different areas on different end users such as the AI researcher and the designer and, finally, outlined the influence of the different game AI areas on the player, the game and their interaction. From the high-level analysis of the game AI field we moved on to the detailed analysis of the game AI areas that compose it and thoroughly surveyed the meaningful interconnections between the different areas.

The total number of strong and weak influences is rather small compared to all possible interconnections between the areas, which clearly signals the research capacity of the game AI field for further explorations. We can distinguish a number of connections which are currently very active, meaning that much work currently goes on in one area that draws on work in another area. Here we see, for example, the connection between AI that plays to win in a general fashion in conjunction with the use of tree search algorithms: the MCTS algorithm was invented in the context of board game-playing, proved to be really useful in the general game playing competition, and is being investigated for use in games as different as *StarCraft* (Blizzard Entertainment, 1998) and *Super Mario Bros* (Nintendo, 1985). Improvements and modifications to the algorithm have been flowing back and forth between the various areas. Another indicative connection that is alive and strong is between player modeling and procedural content generation, where it is now common for newly devised PCG algorithms and experimental studies to include player behavioral or player experience models.

One can also study the currently strong areas by trying to cluster the trending topics in recent iterations of the IEEE CIG and AIIDE conferences. Such studies always include some form of selection bias, as papers can usually be counted into more than one area (e.g., depending on if you group by method or domain), but if you start from the session groupings made by the program chairs of each conference you achieve at least some inter-subjective validity. According to such a clustering, the most active topics over the last few years have been player (or emotion) modeling, game analytics, general game AI, real-time strategy game playing—especially *StarCraft* (Blizzard Entertainment, 1998)—and PCG (in general). Another perspective of the trend in game AI research is the varying percentage of studies on NPC (or game agent) behavior learning over other uses of AI in games at the two key conferences in the field (IEEE CIG and AIIDE). Our preliminary calculations suggest that while, initially, AI was mainly applied for NPC control and for playing board/card games well—more than 75% of CIG and AIIDE papers were linked to NPC behavior and agent game playing in 2005—that trend has drastically changed as entirely new (non-traditional) uses of AI became more common over the years—e.g., roughly 52% of the papers in CIG and AIIDE in 2011 did not involve game agents and NPC behavior. These facts indicate a shift in the use of AI in and for games towards multiple non-traditional applications—which tend to be traditional by now—for the development of better games [764].

But it is maybe even more interesting to look at all those connections that are unexploited or underexploited or potentially strong. For example, player modeling is potentially very important in the development of AI that controls believable, interesting or curious agents, but this has not been explored in enough depth yet; the

same holds for the application of user (or else, designer) modeling principles towards the personalization of AI-assisted game design. Believable agents have, in turn, not been used enough in content generation (either autonomous or assisted). A grand vision of game AI for the years to come is to let it identify its own role within game design and development as it sees fit. In the last chapter of this book we discuss frontier research topics as such and identify unexplored roles of AI in games.

## 6.4 Summary

We hope that with this chapter of this book, we have been able to give our readers a sense of how this—by now rather large and multifaceted—research field hangs together, and what could be done to integrate it further. We realize that this is only our view of its dynamics and interconnections, and that there are (or could be) many competing views. We look forward to seeing those in upcoming studies in the field.

Finally, it is important to note that it would have been impossible to provide a complete survey of all the areas as, first, the game AI field is growing rapidly and, second, it is not the core objective of the book. This means that the bibliography is indicative rather than exhaustive and serves as a general guideline for the reader. The website of the book, instead of the book per se, will be kept up to date regarding important new readings for each area.

The next and final chapter of the book is dedicated to a few long-standing, yet rather unexplored, research frontiers of game AI. We believe that any advances made in these directions will lead to scientific breakthroughs not merely within game AI but largely in both games (their design, technology and analysis) and AI per se.



## Chapter 7

# Frontiers of Game AI Research

In this final chapter of the book we discuss a number of long-term visionary goals of game AI, putting an emphasis on the **generality** of AI and the **extensibility** of its roles within games. In particular, in Section 7.1 we discuss our vision for general behavior for each one of the three main uses of AI in games. Play needs to become general; generators are required to have general generative capacities across games, content types, designers and players; models of players also need to showcase general modeling abilities. In Section 7.2 we also discuss roles of AI that are still unexplored and certainly worth investigating in the future. The book ends with a discussion dedicated to general ethical considerations of game AI (Section 7.3).

### 7.1 General General Game AI

As evidenced from the large volume of studies the game AI research area has been supported by an active and healthy research community for more than a decade—at least since the start of the IEEE CIG and the AIIDE conference series in 2005. Before then, research had been conducted on AI in board games since the dawn of automatic computing. Initially, most of the work published at IEEE CIG or AIIDE was concerned with learning to play a particular game as well as possible, or using search/planning algorithms to play a game as well as possible without learning. Gradually, a number of new applications for AI in games and for games in AI have come to complement the original focus on AI for playing games [764]. Papers on procedural content generation, player modeling, game data mining, human-like playing behavior, automatic game testing and so on have become commonplace within the community. As we saw in the previous chapter there is also a recognition that all these research endeavors depend on each other [785]. However, almost all research projects in the game AI field are very *specific*. Most published papers describe a particular method—or a comparison of two or more methods—for performing a single task (playing, modeling, generating, etc.) in a single game. This is problematic in several ways, both for the scientific value and for the practical appli-

cability of the methods developed and studies made in the field. If an AI approach is only tested on a single task for a single game, how can we argue that is an advance in the scientific study of artificial intelligence? And how can we argue that it is a useful method for a game designer or developer, who is likely working on a completely different game than the one the method was tested on?

As discussed in several parts of this book general game playing is an area that has already been studied extensively and constitutes one of the key areas of game AI [785]. The focus of generality solely on play, however, is *very narrow* as the possible roles of AI and general intelligence in games are many, including game design, content design and player experience design. The richness of the cognitive skills and affective processes required to successfully complete these tasks has so far been largely ignored by game AI research. We thus argue, that while the focus on general AI needs to be retained, research on general game AI needs to expand beyond mere game playing. The new scope for **general general game AI** beyond game-playing broadens the applicability and capacity of AI algorithms and our understanding of intelligence as tested in a creative domain that interweaves problem solving, art, and engineering.

For general game AI to eventually be *truly* general, we argue that we need to extend the generality of general game playing to all other ways in which AI is (or can be) applied to games. More specifically we argue that the field should move towards methods, systems and studies that incorporate three different types of generality:

1. **Game generality.** We should develop AI methods that work with not just one game, but with any game (within a given range) that the method is applied to.
2. **Task generality.** We should develop methods that can do not only one task (playing, modeling, testing, etc) but a number of different, related tasks.
3. **User/designer/player generality.** We should develop methods that can model, respond to and/or reproduce the very large variability among humans in design style, playing style, preferences and abilities.

We further argue that all of this generality can be embodied into the concept of **general game design**, which can be thought of as a final frontier of AI research within games. Further details about the notion of *general* general game AI can be found in the vision paper we co-authored about this frontier research area [718]. It is important to note that we are not arguing that more focused investigations into methods for single tasks in single games are useless; these are often important as proofs-of-concept or industrial applications and they will continue to be important in the future, but there will be an increasing need to validate such case studies in a more general context. We are also not envisioning that everyone will suddenly start working on general methods. Rather, we are positing generalizations as a long-term goal for our entire research community. Finally, the general systems of game AI that we envision ought to have a real-world use. There is a risk that by making systems too general we might end up not finding applications of these general systems to any specific real-world problem. Thus, the system's applicability (or usefulness) sets our core constraint towards this vision of general game AI. More specifically, we envi-

sion *general* general game AI systems that are nevertheless integrated successfully within *specific* game platforms or game engines.

### 7.1.1 General Play

The problem of playing games is the one that has been most generalized so far. There already exist at least three serious benchmarks or competitions attempting to pose the problem of playing games *in general*, each in its own imperfect way. The General Game Playing Competition, often abbreviated GGP [223], the Arcade Learning Environment [40] and the General Video Game AI competition [528]; all three have been discussed in various places in this book. The results from these competitions so far indicate that general purpose search and learning algorithms by far outperform more domain-specific solutions and “clever hacks”. Somewhat simplified, we can say that variations of Monte Carlo tree search perform best on GVGAI and GGP [202], and for ALE (where no forward model is available so learning a policy for each game is necessary) reinforcement learning with deep networks [464] and search-based iterative width [389, 301, 390] perform best. This is a very marked difference from the results of the game-specific competitions, indicating the lack of domain-independent solutions.

While these are each laudable initiatives and currently the focus of much research, in the future we will need to expand the scope of these competitions and benchmarks considerably, including expanding the range of games available to play and the conditions under which gameplay happens. We need game playing benchmarks and competitions capable of expressing any kind of game, including puzzle games, 2D arcade games, text adventures, 3D action-adventures and so on; this is the best way to test general AI capacities and reasoning skills. We also need a number of different ways of interfacing with these games—there is room for both benchmarks that give agents no information beyond the raw screen data but give them hours to learn how to play the game, and those that give agents access to a forward model and perhaps the game code itself, but expect them to play any game presented to them with no time to learn. These different modes test different AI capabilities and tend to privilege different types of algorithms. It is worth noting that the GVGAI competition is currently expanding to different types of playing modes, and has a long-term goal to include many more types of games [527].

We also need to differentiate away from just measuring how to play games optimally. In the past, several competitions have focused on agents that play games in a human-like manner; these competitions have been organized similarly to the classic Turing test [263, 619]. Playing games in a human-like manner is important for a number of reasons, such as being able to test levels and other game content as part of search-based generation, and to demonstrate new content to players. So far, the question of how to play games in a human-like manner *in general* is mostly unexplored; some preliminary work is reported in [337]. Making progress here will likely involve modeling how humans play games in general, including characteris-

tics such as short-term memory, reaction time and perceptual capabilities, and then translating these characteristics to playing style in individual games.

### 7.1.2 General Game Generation and Orchestration

The study of PCG [616] for the design of game levels has reached a certain maturity and is, by far, the most popular domain for the application of PCG algorithms and approaches (e.g., see [720, 785, 783] among many). What is common in most of the content generation studies covered in this book, however, is their *specificity* and strong dependency of the representation chosen on the game genre examined. For the Mario AI Framework, for instance, the focus on a single level generation problem has been very much a mixed blessing: it has allowed for the proliferation and simple comparison of multiple approaches to solving the same problem, but has also led to a clear overfitting of methods. Even though some limited generalization is expected within game levels of the same genre, the level generators that have been explored so far clearly do not have the capacity of *general* level design. We argue that there needs to be a shift in how level generation is viewed. The obvious change of perspective is to create **general level generators**—level generators with general intelligence that can generate levels for any game (within a specified range). That would mean that levels are generated successfully across game genres and players and that the output of the generation process is meaningful and playable as well as entertaining for the player. Further, a general level generator should be able to coordinate the generative process with the other computational game designers who are responsible for the other parts of the game design.

To achieve general level design intelligence algorithms are required to capture as much of the level design space as possible at different representation resolutions. We can think of representation learning approaches such as deep autoencoders [739] capturing core elements of the level design space and fusing various game genres within a sole representation—as already showcased by a few methods, such as the Deep Learning Novelty Explorer [373]. The first attempt to create a benchmark for general level generation has recently been launched in the form of the Level Generation Track of the GVGAI competition. In this competition track, competitors submit level generators capable of generating levels for unseen games. The generators are then supplied with the description of several games, and produce levels which are judged by human judges [338]. Initial results suggest that constructing competent level generators that can produce levels for any game is much more challenging than constructing competent level generators for a single game. A related effort is the Video Game Level Corpus [669] which aims to provide a set of game levels across multiple games and genres which can be used for training level generators for data-driven procedural content generation.

While level generation, as discussed above, is one of the main examples of procedural content generation, there are many other aspects (or **facets**) of games that can be generated. These include visuals, such as textures and images; narrative, such

as quests and backstories; audio, such as sound effects and music; and of course all kinds of things that go into game levels, such as items, weapons, enemies and personalities [381, 616]. However, an even greater challenge is the generation of complete games, including some or all of these facets together with the rules of the game. While, as covered in Chapter 4, there have been several attempts to generate games (including their rules) we are not aware of any approach to generating games that tries to generate more than two of the facets of games listed above. We are also not aware of any game generation system that even tries to generate games of more than one genre. Multi-faceted generation systems like *Sonancia* [394, 395] co-generate horror game levels with corresponding soundscapes but do not cater to the generation of rules. It is clear that the very domain-limited and facet-limited aspects of current game generation systems result from intentionally limiting design choices in order to make the very difficult problem of generating complete games tractable. Yet, in order to move beyond what could be argued to be toy domains and start to fulfill the promise of game generation, we need systems that can generate multiple facets of games at the same time, and that can generate games of different kinds.

This process has been defined as facet (domain) **orchestration** in games [371, 324]. Orchestration refers to the process of *harmonizing game generation*. Evidently, orchestration is a necessary process when we consider the output of *two or more* content type generators—such as visuals and audio—for the generation of a complete game. Drawing inspiration from music, orchestration may vary from a *top-down*, conductor-driven process to a *bottom-up*, free-from generation process [371]. A few years ago, something very much like general game generation and orchestration was outlined as the challenges of “multi-content, multi-domain PCG” and “generating complete games” [702]. It is interesting to note that there has not seemingly been any attempt to create more general game generators since then, perhaps due to the complexity of the task. A recent study by Karavolos et al. [324] moves towards the orchestration direction as it fuses level and game design parameters in first-person shooters via deep convolutional neural networks. The trained networks can be used to generate balanced games. Currently the only genre for which generators have been built that can generate high-quality (complete) games is abstract board games. Once more genres have been “conquered”, we hope that the task of building more general level generators can begin.

Linked to the tasks of orchestration and general game generation there are important questions with respect to the **creative** capacity of the generation process that remain largely unanswered. For example, how creative can a generator be and how can we assess it? Is it, for instance, deemed to have appreciation, skill, and imagination [130]? When it comes to the evaluation of the creative capacity of current PCG algorithms a case can be made that most of them possess only skill. Does the creator manage to explore novel combinations within a constrained space, thereby resulting in **exploratory** game design creativity [53]; or, is on the other hand trying to break existing boundaries and constraints within game design to come up with entirely new designs, demonstrating **transformational** creativity [53]? If used in a mixed-initiative fashion, does it enhance the designer’s creativity by boosting the possi-

bility space for her? Arguably, the appropriateness of various evaluation methods for autonomous PCG creation or mixed-initiative co-creation [774] remains largely unexplored within both human and computational creativity research.

### 7.1.3 General Game Affective Loop

It stands to reason that general intelligence implies (and is tightly coupled with) general emotional intelligence [443]. The ability to recognize human behavior and emotion is a complex yet critical task for human communication that acts as a facilitator of general intelligence [157]. Throughout evolution, we have developed particular forms of advanced cognitive, emotive and social skills to address this challenge. Beyond these skills, we also have the capacity to detect affective patterns across people with different moods, cultural backgrounds and personalities. This generalization ability also extends, to a degree, across contexts and social settings. Despite their importance, the characteristics of social intelligence have not yet been transferred to AI in the form of general emotive, cognitive or behavioral models. While research in affective computing [530] has reached important milestones such as the capacity for real-time emotion recognition [794]—which can be faster than humans under particular conditions—all key findings suggest that any success of affective computing is heavily dependent on the domain, the task at hand, and the context in general. This *specificity* limitation is particularly evident in the domain of games [781] as most work in modeling player experience focuses on particular games, under well-controlled conditions with particular, small sets of players (see [783, 609, 610, 435] among many). In this section we identify and discuss two core unexplored and interwoven aspects of modeling players that are both important and necessary steps towards the long-term aim of game AI to realize truly adaptive games. The first aspect is the closure of the affective loop in games; the second aspect is the construction of general models capable of capturing experience across players and games.

As stated at the start of this book, affective computing is best realized within games in what we name the **game affective loop**. While the phases of emotion elicitation, affect modeling and affect expression have offered some robust solutions by now, the very loop of affective-based interaction has not been closed yet. Aside from a few studies demonstrating some affect-enabled adaptation of the game [772, 617] the area remains largely unexplored. It is not only the complexity of modeling players and their experience that is the main hurdle against any advancement. What is also far from trivial is the appropriate and meaningful integration of any of these models in a game. The questions of how often the system should adapt, what it should alter and by what degree are not easy to answer. As most of the questions are still open to the research community the only way to move forward is to do more research in adaptive games involving affective aspects of the experience. Existing commercial-standard games that already realize the affective loop such as *Nevermind* (Flying Mollusk, 2016) are the ambassadors for further work in this area.

Once the game affective loop is successfully realized within particular games the next goal for game AI is the **generality of affect-based interaction** across games. The game affective loop should not only be operational; it should ideally be general too. For AI in games to be general beyond game-playing it needs to be able to recognize general emotional and cognitive-behavioral patterns. This is essentially AI that can detect context-free emotive and cognitive reactions and expressions across contexts and builds general computational models of human behavior and experience which are grounded in a general gold standard of human behavior. So far we have only seen a few proof-of-concept studies in this direction. Early work within the game AI field focused on the ad-hoc design of general metrics of player interest that were tested across different prey-predator games [768, 767]. In other, more recent, studies predictors of player experience were tested for their ability to capture player experience across dissimilar games [431, 612, 97]. Another study on deep multimodal fusion can be seen as an embryo for further research in this direction [435], in which various modalities of player input such as player metrics, skin conductance and heart activity have been fused using stacked autoencoders. Discovering entirely new representations of player behavior and emotive manifestations across games, modalities of data, and player types is a first step towards achieving general player modeling. Such representations can, in turn, be used as the basis for approximating the *ground truth* of user experience in games.

## 7.2 AI in Other Roles in Games

The structure of this book reflects our belief that playing games, generating content and modeling players are the central applications of AI methods in games. However, there are many variants and use cases of game playing, player modeling or content generation that we have not had time to explore properly in the book, and which in some cases not have been explored in the literature at all. Further, there are some applications of AI in games that cannot be really classified as special cases of our “big three” AI applications in games, despite our best efforts. This section briefly sketches some of these applications, some of which may be important future research directions.

**Playtesting:** One of the many use cases for AI for playing games is to test the games. Testing games for bugs, balancing player experience and behavior, and other issues is important in game development, and one of the areas where game developers are looking for AI assistance. While playtesting is one of the AI capabilities within many of the mixed-initiative tools discussed in Chapter 4, there has also been work on AI-based playtesting outside of that context. For example, Denzinger et al. evolved action sequences to find exploits in sports games, with discouragingly good results [165]. For the particular case of finding bugs and exploits in games, one of the research challenges is to find a good and representative coverage of problems, so as to deliver an accurate picture to the development team of how many problems

there are and how easy they are to run into, and allow prioritization of which problems to fix.

**Critiquing Games:** Can AI methods meaningfully judge and critique games? Game criticism is hard and generally depends on deep understanding of not only games but also the surrounding cultural context. Still, there might be automated metrics that are useful for game criticism, and can provide information to help reviewers, game curators and others in selecting which games to consider for reviewing for inclusion in app stores. The ANGELINA game generation system is one of the few examples towards this direction [136] in which AI generates the overview of the game to be played.

**Hyper-formalist Game Studies:** AI methods can be applied to corpora of games in order to understand distributions of game characteristics. For example, decision trees can be used to visualize patterns of resource systems in games [312]. There are likely many other ways of using game AI for game studies that are still to be discovered.

**Game Directing:** The outstanding feature of *Left 4 Dead* (Valve Corporation, 2008) was its AI director, which adjusted the onslaught of zombies to provide a dramatic curve of challenge for players. While simple and literally one-dimensional (only a single dimension of player experience was tracked), the AI director proved highly effective. There is much room for creating more sophisticated AI directors; the experience-driven PCG framework [783] is one potential way within which to work towards this.

**Creative Inspiration:** While designing a complete game that actually works likely requires a very complex generator, it can be simpler to generate an idea for new games, that are then designed by humans. Creative ideation tools range from simple word-recombination-based tools implemented as card games or Twitter bots, to elaborate computational creativity systems such as the *What If-Machine* [391].

**Chat Monitoring:** In-game chats are important in many online multi-player games, as they allow people to collaborate within games and socialize through them. Unfortunately, such chats can also be used to threaten or abuse other players. Given the very large volume of chat messages sent through a successful online game, it becomes impossible for the game developers to curate chats manually. In the efforts to combat toxic behavior, some game developers have therefore turned to machine learning. Notably, Riot Games have trained algorithms to recognize and remove toxic behavior in the MOBA *League of Legends* (Riot Games, 2009) [413]. Even worse, sexual predation can be seen in some games, where pedophiles use game chats to reach children; there have been attempts to use machine learning to detect sexual predators in game chats too [241].

**AI-Based Game Design:** Throughout most of the book, we have assumed the ex-

istence of a game or at least a game design, and discussed how AI can be used to play that game, generate content for it or model its players. However, one could also start from some AI method or capability and try to design a game that builds on that method or capability. This could be seen as an opportunity to showcase AI methods in the context of games, but it could also be seen as a way of advancing game design. Most classic game designs originate in an era where there were few effective AI algorithms, there was little knowledge among game designers about those AI algorithms that existed, and CPU and memory capacity of home computers was too limited to allow anything beyond simple heuristic AI and some best-first search to be used. One could even say that many classic video game designs are an attempt to design around the lack of AI—for example, the lack of good dialog AI for NPCs led to the use of dialog trees, the lack of AIs that could play FPS games believably and competently led to FPS game designs where most enemies are only on-screen for a few seconds so that you do not notice their lack of smarts, and the lack of level generation methods that guaranteed balance and playability led to game designs where levels did not need to be completable. The persistence of such design patterns may be responsible for the relatively low utilization of interesting AI methods within commercial game development. By starting with the AI and designing a game around it, new design patterns that actually exploit some of the recent AI advances can be found.

Several games have been developed within the game AI research community specifically to showcase AI capabilities, some of which have been discussed in this book. Three of the more prominent examples are based on Stanley et al.’s work on neuroevolution and the NEAT algorithm: *NERO*, which is an RTS-like game where the player trains an army through building a training environment rather than controlling it directly [654]; *Galactic Arms Race*, in which weapons controlled through neural networks are indirectly collectively evolved by thousands of players [250, 249]; and *Petalz*, which is a Facebook game about collecting flowers based on a similar idea of selection-based collective neuroevolution [565, 566]. Other games have been built to demonstrate various adaptation mechanisms, such as *Infinite Tower Defense* [25] and *Maze-Ball* [780]. Within interactive narrative it is relatively common to build games that showcase specific theories and methods; a famous example is *Façade* [441] and another prominent example is *Prom Week* [447]. Treanor et al. have attempted to identify AI-based game design patterns, and found a diverse array of roles in which AI can be or has been used in games, and a number of avenues for future AI-based game design [724].

### 7.3 Ethical Considerations

Like all technologies, artificial intelligence, including game AI, can be used for many purposes, some of them nefarious. Perhaps even more importantly, technology can have ethically negative or at least questionable effects even when there is no malicious intent. The ethical effects of using AI with and in games are not always

obvious, and the topic is not receiving the attention it should. This short section looks at some of the ways in which game AI intersects with ethical questions. For general AI research issues, ethics and values we refer the interested reader to the Asilomar AI Principles<sup>1</sup> developed in conjunction with the 2017 Asilomar conference.

Player modeling is perhaps the part of game AI where the ethical questions are most direct, and perhaps most urgent. There is now a vigorous debate about the mass collection of data about us both by government entities (such as the US National Security Agency or the United Kingdom's GCHQ) and private entities (such as Google, Amazon, Facebook and Microsoft) [64, 502]. With methodological advances in data mining, it is becoming possible to learn more and more about individual people from their digital traces, including inferring sensitive information and predicting behavior. Given that player modeling involves large-scale data collection and mining, many of the same ethical challenges exist in player modeling as in the mining of data about humans in general. Mikkelsen et al. present an overview of ethical challenges for player modeling [458]. Below we give some examples of such challenges.

**Privacy:** It is becoming increasingly possible and even practicable to infer various real-life traits and properties of people from their in-game behavior. This can be done without the consent or even knowledge of the subject, and some of the information can be of a private and sensitive nature. For example, Yee and colleagues investigated how player choices in *World of Warcraft* (Blizzard Entertainment, 2004) correlated with the personalities of players. They used data about players' characters from the Armory database of *World of Warcraft* (Blizzard Entertainment, 2004) and correlated this information with personality tests administered to players; multiple strong correlations were found [788]. In a similar vein, a study investigated how players' life motives correlated with their *Minecraft* (Mojang, 2011) log files [101]. That research used the life motivation questionnaires of Steven Reiss, and found that players' self-reported life motives (independence, family, etc.) were expressed in a multitude of ways inside constructed *Minecraft* (Mojang, 2011) worlds. Using a very different type of game strong correlations have been found between playing style in the first-person shooter *Battlefield 3* (Electronic Arts, 2011) and player characteristics such as personality [687], age [686] and nationality [46]. It is entirely plausible that similar methods could be used to infer sexual preferences, political views, health status and religious beliefs. Such information could be used by advertising networks to serve targeted ads, by criminals looking to blackmail the player, by insurance companies looking to differentiate premiums, or by malevolent political regimes for various forms of suppression. We do not know yet what can be predicted and with what accuracy, but it is imperative that more research be done on this within the publicly available literature; it is clear that this kind of research will also be carried out behind locked doors.

---

<sup>1</sup> <https://futureoflife.org/ai-principles/>

**Ownership of Data:** Some player data can be used to recreate aspects of the player’s behavior; this is the case for e.g., the *Drivatars* in Microsoft’s *Forza Motorsport* series, and more generally for agents created according to the procedural persona concept [267]. It is currently not clear who owns this data, and if the game developer/publisher owns the data, what they can do with it. Will the game allow other people to play against a model of you, i.e., how you would have played the game? If so, can it identify you to other players as the origin of this data? Does it have to be faithful to the behavioral model of you, or can it add or distort aspects of your playing behavior?

**Adaptation:** Much of the research within game AI is concerned with adaptation of games, with the experience-driven PCG framework being the perhaps most complete account on how to combine player modeling with procedural content generation to create personalized game experiences [783]. However, it is not clear that it is always a good thing to adapt games to players. The “filter bubble” is a concept within discussion of social networks which refers to the phenomenon where collaborative filtering ensures that users are only provided with content that is already in line with their political, ethical, or aesthetic preferences, leading to a lack of healthy engagement with other perspectives. Excessive adaptation and personalization might have a similar effect, where players are funneled into a narrow set of game experiences.

**Stereotypes:** Anytime we train a model using some dataset, we run the risk of reproducing stereotypes within that dataset. For example, it has been shown that word embeddings trained on standard datasets of the English language reproduce gender-based stereotypes [93]. The same effects could be present when modeling player preferences and behavior, and the model might learn to reproduce prejudiced conceptions regarding gender, race, etc. Such problems can be exacerbated or ameliorated by the tools made available to players for expressing themselves in-game. For example, Lim and Harrell have developed quantitative methods for measuring and addressing bias in character creation tools [386].

**Censorship:** Of course, it is entirely possible, and advisable, to use AI methods to promote ethical behavior and uphold ethical values. Earlier, Section 7.2 discussed the examples of AI for filtering player chats in online multi-player games, and for detecting sexual predators. While such technologies are generally welcome, there are important ethical considerations in how they should be deployed. For example, a model that has been trained to recognize hate speech might also react to normal in-game jargon; setting the right decision threshold might involve a delicate tradeoff between ensuring a welcoming game environment and not restricting communications unduly.

**AI Beyond Games:** Finally, a somewhat more far-fetched concern, but one we believe still merits discussion is the following. Games are frequently used to train and test AI algorithms—this is the main aim of, for example, the General Video Game

AI Competition and the Arcade Learning Environment. However, given how many games are focused on violent competition, does this mean that we focus unduly on the development of violence in artificial intelligence? What effects could this have on AI that is trained on games but employed in other domains, such as transport or health care?

## 7.4 Summary

In this last chapter of this book we went through directions that we view as critical and important for the advancement of the game AI field. Initially we have argued that the general intelligence capacity of machines needs to be both explored and exploited to its full potential (1) across the different **tasks** that exist within the game design and development process, including but absolutely no longer limited to game playing; (2) across different **games** within the game design space; and (3) across different **users** (players or designers) of AI. We claim that, thus far, we have underestimated the potential for general AI within games. We also claim that the currently dominant practice of only designing AI for a specific task within a specific domain will eventually be detrimental to game AI research as algorithms, methods and epistemological procedures will remain specific to the task at hand. As a result, we will not manage to push the boundaries of AI and exploit its full capacity for game design. We are inspired by the general game-playing paradigm and the recent successes of AI algorithms in that domain and suggest that we become less specific about all subareas of the game AI field including player modeling and game generation. Doing so would allow us to detect and mimic different general cognitive and emotive skills of humans when designing games. It is worth noting, again, that we are not advocating that all research within the game AI field focuses on generality right now; studies on particular games and particular tasks are still valuable, given how little we still understand and can do. But over time, we predict that more and more research will focus on generality across tasks, games and users, because it is in the general problems that the interesting research questions of the future lie. It seems that we are not alone in seeing this need as other researchers have argued for the use of various game-related tasks (not just game playing) to be used in artificial general intelligence research [799].

The path towards achieving general game artificial intelligence is still largely unexplored. For AI to become less specific—yet remain relevant and useful for game design—we envision a number of immediate steps that could be taken: first and foremost, the game AI community needs to adopt an **open-source** accessible strategy so that methods and algorithms developed across the different tasks are shared among researchers for the advancement of this research area. Venues such as the current game AI research portal<sup>2</sup> could be expanded and used to host successful methods and algorithms. For the algorithms and methods to be of direct use particular tech-

---

<sup>2</sup> <http://www.aigameresearch.org/>

nical specifications need to be established—e.g., such as those established within game-based AI benchmarks—which will maximize the interoperability among the various tools and elements submitted. Examples of benchmarked specifications for the purpose of general game AI research include the general video game description language and the puzzle game engine PuzzleScript.<sup>3</sup> Finally, following the GVGAI competition paradigm, we envision a new set of competitions rewarding general player models, AI-assisted tools and game generation techniques. These competitions would further motivate researchers to work in this exciting research area and enrich the database of open-access interoperable methods and algorithms, directly contributing to the state of the art in computational general game design.

Beyond generality we also put a focus on the extensibility of AI roles within games. In that regard, we outlined a number of AI roles that are underrepresented currently but nevertheless define very promising research frontiers for game AI. These include the roles of AI as playtester, game critic, game studies formalist, director, creative designer, and gameplay ethics judge. Further, we view the placement of AI at the very center of the design process (AI-based game design) as another critical research frontier.

This chapter, and the book itself, concluded with a discussion on the ethical implications of whatever we do in game AI research. In particular, we discussed aspects such as the privacy and ownership of data, the considerations about game adaptation, the emergence of stereotypes through computational models of players, the risks of AI acting as a censor, and finally the ethical constraints imposed on AI by “unethical” aspects of the very nature of games.

---

<sup>3</sup> <http://www.puzzlescript.net/>



## References

1. Espen Aarseth. Genre trouble. *Electronic Book Review*, 3, 2004.
2. Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. TensorFlow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.
3. Ryan Abela, Antonios Liapis, and Georgios N. Yannakakis. A constructive approach for the generation of underwater environments. In *Proceedings of the FDG workshop on Procedural Content Generation in Games*, 2015.
4. David H. Ackley, Geoffrey E. Hinton, and Terrence J. Sejnowski. A learning algorithm for Boltzmann machines. *Cognitive Science*, 9(1):147–169, 1985.
5. Alexandros Agapitos, Julian Togelius, Simon M. Lucas, Jürgen Schmidhuber, and Andreas Konstantinidis. Generating diverse opponents with multiobjective evolution. In *Computational Intelligence and Games, 2008. CIG'08. IEEE Symposium On*, pages 135–142. IEEE, 2008.
6. Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. In *ACM SIGMOD Record*, pages 207–216. ACM, 1993.
7. Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. In *Proceedings of the 20th International Conference on Very Large Data Bases, VLDB*, pages 487–499, 1994.
8. John B. Ahlquist and Jeannie Novak. *Game development essentials: Game artificial intelligence*. Delmar Pub, 2008.
9. Zach Aikman. Galak-Z: Forever: Building Space-Dungeons Organically. In *Game Developers Conference*, 2015.
10. Bob Alexander. The beauty of response curves. *AI Game Programming Wisdom*, page 78, 2002.
11. Krishna Aluru, Stefanie Tellex, John Oberlin, and James MacGlashan. Minecraft as an experimental world for AI in robotics. In *AAAI Fall Symposium*, 2015.
12. Samuel Alvernaz and Julian Togelius. Autoencoder-augmented neuroevolution for visual doom playing. In *IEEE Conference on Computational Intelligence and Games*. IEEE, 2017.
13. Omar Alzoubi, Rafael A. Calvo, and Ronald H. Stevens. Classification of EEG for Affect Recognition: An Adaptive Approach. In *AI 2009: Advances in Artificial Intelligence*, pages 52–61. Springer, 2009.
14. Mike Ambinder. Biofeedback in gameplay: How Valve measures physiology to enhance gaming experience. In *Game Developers Conference*, San Francisco, California, US, 2011.
15. Dan Amerson, Shaun Kime, and R. Michael Young. Real-time cinematic camera control for interactive narratives. In *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology*, pages 369–369. ACM, 2005.

16. Elisabeth André, Martin Klesen, Patrick Gebhard, Steve Allen, and Thomas Rist. Integrating models of personality and emotions into lifelike characters. In *Affective interactions*, pages 150–165. Springer, 2000.
17. John L. Andreassi. *Psychophysiology: Human Behavior and Physiological Response*. Psychology Press, 2000.
18. Rudolf Arnheim. *Art and visual perception: A psychology of the creative eye*. University of California Press, 1956.
19. Ivon Arroyo, David G. Cooper, Winslow Burleson, Beverly Park Woolf, Kasia Muldner, and Robert Christoperson. Emotion sensors go to school. In *Proceedings of Conference on Artificial Intelligence in Education (AIED)*, pages 17–24. IOS Press, 2009.
20. W. Ross Ashby. Principles of the self-organizing system. In *Facets of Systems Science*, pages 521–536. Springer, 1991.
21. Daniel Ashlock. *Evolutionary computation for modeling and optimization*. Springer, 2006.
22. Stylianos Asteriadis, Kostas Karpouzis, Noor Shaker, and Georgios N. Yannakakis. Does your profile say it all? Using demographics to predict expressive head movement during gameplay. In *Proceedings of UMAP Workshops*, 2012.
23. Stylianos Asteriadis, Paraskevi Tzouveli, Kostas Karpouzis, and Stefanos Kollias. Estimation of behavioral user state based on eye gaze and head pose—application in an e-learning environment. *Multimedia Tools and Applications*, 41(3):469–493, 2009.
24. Phillipa Avery, Sushil Louis, and Benjamin Avery. Evolving coordinated spatial tactics for autonomous entities using influence maps. In *Computational Intelligence and Games, 2009. CIG 2009. IEEE Symposium on*, pages 341–348. IEEE, 2009.
25. Phillipa Avery, Julian Togelius, Elvis Alistar, and Robert Pieter van Leeuwen. Computational intelligence and tower defence games. In *Evolutionary Computation (CEC), 2011 IEEE Congress on*, pages 1084–1091. IEEE, 2011.
26. Ruth Aylett, Sandy Louchart, Joao Dias, Ana Paiva, and Marco Vala. FearNot!—an experiment in emergent narrative. In *Intelligent Virtual Agents*, pages 305–316. Springer, 2005.
27. Simon E. Ortiz B., Koichi Moriyama, Ken-ichi Fukui, Satoshi Kurihara, and Masayuki Nu-mao. Three-subagent adapting architecture for fighting videogames. In *Pacific Rim International Conference on Artificial Intelligence*, pages 649–654. Springer, 2010.
28. Sander Bakkes, Pieter Spronck, and Jaap van den Herik. Rapid and reliable adaptation of video game AI. *IEEE Transactions on Computational Intelligence and AI in Games*, 1(2):93–104, 2009.
29. Sander Bakkes, Shimon Whiteson, Guangliang Li, George Viorel Vișniuc, Efstrathios Charitos, Norbert Heijne, and Arjen Swellengrebel. Challenge balancing for personalised game spaces. In *Games Media Entertainment (GEM), 2014 IEEE*, pages 1–8. IEEE, 2014.
30. Rainer Banse and Klaus R. Scherer. Acoustic profiles in vocal emotion expression. *Journal of Personality and Social Psychology*, 70(3):614, 1996.
31. Ray Barrera, Aung Sithu Kyaw, Clifford Peters, and Thet Naing Swe. *Unity AI Game Programming*. Packt Publishing Ltd, 2015.
32. Gabriella A. B. Barros, Antonios Liapis, and Julian Togelius. Data adventures. In *Proceedings of the FDG workshop on Procedural Content Generation in Games*, 2015.
33. Richard A. Bartle. *Designing virtual worlds*. New Riders, 2004.
34. Chris Bateman and Richard Boon. *21st Century Game Design (Game Development Series)*. Charles River Media, Inc., 2005.
35. Chris Bateman and Lennart E. Nacke. The neurobiology of play. In *Proceedings of the International Academic Conference on the Future of Game Design and Technology*, pages 1–8. ACM, 2010.
36. Christian Bauckhage, Anders Drachen, and Rafet Sifa. Clustering game behavior data. *IEEE Transactions on Computational Intelligence and AI in Games*, 7(3):266–278, 2015.
37. Yoann Baveye, Jean-Noël Bettinelli, Emmanuel Dellandrea, Liming Chen, and Christel Chamaret. A large video database for computational models of induced emotion. In *Proceedings of Affective Computing and Intelligent Interaction*, pages 13–18, 2013.
38. Jessica D. Bayliss. Teaching game AI through Minecraft mods. In *2012 IEEE International Games Innovation Conference (IGIC)*, pages 1–4. IEEE, 2012.

39. Farès Belhadj. Terrain modeling: a constrained fractal model. In *Proceedings of the 5th international conference on Computer graphics, virtual reality, visualisation and interaction in Africa*, pages 197–204. ACM, 2007.
40. Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *arXiv preprint arXiv:1207.4708*, 2012.
41. Yoshua Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009.
42. José Luis Bernier, C. Ilia Herráiz, J. J. Merelo, S. Olmeda, and Alberto Prieto. Solving Mastermind using GAs and simulated annealing: a case of dynamic constraint optimization. In *Parallel Problem Solving from Nature (PPSN) IV*, pages 553–563. Springer, 1996.
43. Kent C. Berridge. Pleasures of the brain. *Brain and Cognition*, 52(1):106–128, 2003.
44. Dimitri P. Bertsekas. *Dynamic programming and optimal control*. Athena Scientific Belmont, MA, 1995.
45. Nadav Bhotker, Shai Rozenberg, and Itay Hubara. Playing SNES in the Retro Learning Environment. *arXiv preprint arXiv:1611.02205*, 2016.
46. Mateusz Bialas, Shoshannah Tekofsky, and Pieter Spronck. Cultural influences on play style. In *Computational Intelligence and Games (CIG), 2014 IEEE Conference on*, pages 1–7. IEEE, 2014.
47. Nadia Bianchi-Berthouze and Christine L. Lisetti. Modeling multimodal expression of user’s affective subjective experience. *User Modeling and User-Adapted Interaction*, 12(1):49–84, 2002.
48. Darse Billings, Denis Papp, Jonathan Schaeffer, and Duane Szafron. Opponent modeling in poker. In *AAAI/IAAI*, pages 493–499, 1998.
49. Christopher M. Bishop. *Pattern Recognition and Machine Learning*. 2006.
50. Staffan Björk and Jesper Juul. Zero-player games. In *Philosophy of Computer Games Conference, Madrid*, 2012.
51. Vikki Blake. Minecraft Has 55 Million Monthly Players, 122 Million Sales. *Imagine Games Network*, February 2017.
52. Paris Mavromoustakos Blom, Sander Bakkes, Chek Tien Tan, Shimon Whiteson, Diederik M. Roijers, Roberto Valenti, and Theo Gevers. Towards Personalised Gaming via Facial Expression Recognition. In *Proceedings of AIIDE*, 2014.
53. Margaret A. Boden. What is creativity. *Dimensions of creativity*, pages 75–117, 1994.
54. Margaret A. Boden. Creativity and artificial intelligence. *Artificial Intelligence*, 103(1):347–356, 1998.
55. Margaret A. Boden. *The creative mind: Myths and mechanisms*. Psychology Press, 2004.
56. Slawomir Bojarski and Clare Bates Congdon. REALM: A rule-based evolutionary computation agent that learns to play Mario. In *Computational Intelligence and Games (CIG), 2010 IEEE Symposium on*, pages 83–90. IEEE, 2010.
57. Luuk Bom, Ruud Henken, and Marco Wiering. Reinforcement learning to train Ms. Pac-Man using higher-order action-relative inputs. In *Adaptive Dynamic Programming and Reinforcement Learning (ADPRL), 2013 IEEE Symposium on*, pages 156–163. IEEE, 2013.
58. Blai Bonet and Héctor Geffner. Planning as heuristic search. *Artificial Intelligence*, 129(1–2):5–33, 2001.
59. Philip Bontrager, Ahmed Khalifa, Andre Mendes, and Julian Togelius. Matching games and algorithms for general video game playing. In *Twelfth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2016.
60. Michael Booth. The AI systems of Left 4 Dead. In *Fifth Artificial Intelligence and Interactive Digital Entertainment Conference (Keynote)*, 2009.
61. Adi Botea, Martin Müller, and Jonathan Schaeffer. Near optimal hierarchical path-finding. *Journal of Game Development*, 1(1):7–28, 2004.
62. David M. Bourg and Glenn Seemann. *AI for game developers*. O’Reilly Media, Inc., 2004.
63. Michael Bowling, Neil Burch, Michael Johanson, and Oskari Tammelin. Heads-up limit holdem poker is solved. *Science*, 347(6218):145–149, 2015.

64. Danah Boyd and Kate Crawford. Six provocations for big data. In *A decade in internet time: Symposium on the dynamics of the internet and society*. Oxford Internet Institute, Oxford, 2011.
65. S. R. K. Branavan, David Silver, and Regina Barzilay. Learning to win by reading manuals in a Monte-Carlo framework. *Journal of Artificial Intelligence Research*, 43:661–704, 2012.
66. Michael E. Bratman, David J. Israel, and Martha E. Pollack. Plans and resource-bounded practical reasoning. *Computational Intelligence*, 4(3):349–355, 1988.
67. Leo Breiman, Jerome Friedman, Charles J. Stone, and Richard A. Olshen. *Classification and regression trees*. CRC Press, 1984.
68. Daniel Brewer. Tactical pathfinding on a navmesh. *Game AI Pro: Collected Wisdom of Game AI Professionals*, page 361, 2013.
69. Gerhard Brewka, Thomas Eiter, and Mirosław Truszczyński. Answer set programming at a glance. *Communications of the ACM*, 54(12):92–103, 2011.
70. Rodney Brooks. A robust layered control system for a mobile robot. *IEEE Journal on Robotics and Automation*, 2(1):14–23, 1986.
71. David S. Broomhead and David Lowe. Radial basis functions, multi-variable functional interpolation and adaptive networks. *Royals Signals & Radar Establishment*, 1988.
72. Anna Brown and Alberto Maydeu-Olivares. How IRT can solve problems of ipsative data in forced-choice questionnaires. *Psychological Methods*, 18(1):36, 2013.
73. Daniel Lankford Brown. Mezzo: An adaptive, real-time composition program for game soundtracks. In *Eighth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2012.
74. Cameron Browne. *Automatic generation and evaluation of recombination games*. PhD thesis, Queensland University of Technology, 2008.
75. Cameron Browne. Yavalath. In *Evolutionary Game Design*, pages 75–85. Springer, 2011.
76. Cameron Browne and Frederic Maire. Evolutionary game design. *IEEE Transactions on Computational Intelligence and AI in Games*, 2(1):1–16, 2010.
77. Cameron B. Browne, Edward Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of Monte Carlo tree search methods. *Computational Intelligence and AI in Games, IEEE Transactions on*, 4(1):1–43, 2012.
78. Nicholas J. Bryan, Gautham J. Mysore, and Ge Wang. ISSE: An Interactive Source Separation Editor. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 257–266, 2014.
79. Bobby D. Bryant and Risto Miikkulainen. Evolving stochastic controller networks for intelligent game agents. In *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, pages 1007–1014. IEEE, 2006.
80. Mat Buckland. *Programming game AI by example*. Jones & Bartlett Learning, 2005.
81. Mat Buckland and Mark Collins. *AI techniques for game programming*. Premier Press, 2002.
82. Vadim Bulitko, Yngvi Björnsson, Nathan R. Sturtevant, and Ramon Lawrence. Real-time heuristic search for pathfinding in video games. In *Artificial Intelligence for Computer Games*, pages 1–30. Springer, 2011.
83. Vadim Bulitko, Greg Lee, Sergio Poo Hernandez, Alejandro Ramirez, and David Thue. Techniques for AI-Driven Experience Management in Interactive Narratives. In *Game AI Pro 2: Collected Wisdom of Game AI Professionals*, pages 523–534. AK Peters/CRC Press, 2015.
84. Paolo Burelli. Virtual cinematography in games: investigating the impact on player experience. *Foundations of Digital Games*, 2013.
85. Paolo Burelli and Georgios N. Yannakakis. Combining Local and Global Optimisation for Virtual Camera Control. In *Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games*, Copenhagen, Denmark, August 2010. IEEE.
86. Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data mining and Knowledge Discovery*, 2(2):121–167, 1998.
87. Michael Buro and David Churchill. Real-time strategy game competitions. *AI Magazine*, 33(3):106, 2012.

88. Carlos Busso, Zhigang Deng, Serdar Yildirim, Murtaza Bulut, Chul Min Lee, Abe Kazemzadeh, Sungbok Lee, Ulrich Neumann, and Shrikanth Narayanan. Analysis of emotion recognition using facial expressions, speech and multimodal information. In *Proceedings of the International Conference on Multimodal Interfaces (ICMI)*, pages 205–211. ACM, 2004.
89. Eric Butler, Adam M. Smith, Yun-En Liu, and Zoran Popovic. A mixed-initiative tool for designing level progressions in games. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*, pages 377–386. ACM, 2013.
90. Martin V. Butz and Thies D. Lonneker. Optimized sensory-motor couplings plus strategy extensions for the TORCS car racing challenge. In *IEEE Symposium on Computational Intelligence and Games*, pages 317–324. IEEE, 2009.
91. John T. Cacioppo, Gary G. Berntson, Jeff T. Larsen, Kirsten M. Poehlmann, and Tiffany A. Ito. The psychophysiology of emotion. *Handbook of emotions*, 2:173–191, 2000.
92. Francesco Calimeri, Michael Fink, Stefano Germano, Andreas Humenberger, Giovambattista Ianni, Christoph Redl, Daria Stepanova, Andrea Tucci, and Anton Wimmer. Angry-HEX: an artificial player for Angry Birds based on declarative knowledge bases. *IEEE Transactions on Computational Intelligence and AI in Games*, 8(2):128–139, 2016.
93. Aylin Caliskan, Joanna J. Bryson, and Arvind Narayanan. Semantics derived automatically from language corpora contain human-like biases. *Science*, 356(6334):183–186, 2017.
94. Gordon Calleja. *In-game: from immersion to incorporation*. MIT Press, 2011.
95. Rafael Calvo, Iain Brown, and Steve Scheding. Effect of experimental factors on the recognition of affective mental states through physiological measures. In *AI 2009: Advances in Artificial Intelligence*, pages 62–70. Springer, 2009.
96. Elizabeth Camilleri, Georgios N. Yannakakis, and Alexiei Dingli. Platformer Level Design for Player Believability. In *IEEE Computational Intelligence and Games Conference*. IEEE, 2016.
97. Elizabeth Camilleri, Georgios N. Yannakakis, and Antonios Liapis. Towards General Models of Player Affect. In *Affective Computing and Intelligent Interaction (ACII), 2017 International Conference on*, 2017.
98. Murray Campbell, A. Joseph Hoane, and Feng-hsiung Hsu. Deep blue. *Artificial intelligence*, 134(1-2):57–83, 2002.
99. Henrique Campos, Joana Campos, João Cabral, Carlos Martinho, Jeppe Herlev Nielsen, and Ana Paiva. My Dream Theatre. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-Agent Systems*, pages 1357–1358. International Foundation for Autonomous Agents and Multiagent Systems, 2013.
100. Joana Campos, Carlos Martinho, Gordon Ingram, Asimina Vasalou, and Ana Paiva. My dream theatre: Putting conflict on center stage. In *FDG*, pages 283–290, 2013.
101. Alessandro Canossa, Josep B. Martinez, and Julian Togelius. Give me a reason to dig Minecraft and psychology of motivation. In *Computational Intelligence in Games (CIG), 2013 IEEE Conference on*. IEEE, 2013.
102. Luigi Cardamone, Daniele Loiacono, and Pier Luca Lanzi. Interactive evolution for the procedural generation of tracks in a high-end racing game. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, pages 395–402. ACM, 2011.
103. Luigi Cardamone, Georgios N. Yannakakis, Julian Togelius, and Pier Luca Lanzi. Evolving interesting maps for a first person shooter. In *Applications of Evolutionary Computation*, pages 63–72. Springer, 2011.
104. Justine Cassell. *Embodied conversational agents*. MIT Press, 2000.
105. Justine Cassell, Timothy Bickmore, Mark Billinghurst, Lee Campbell, Kenny Chang, Hannes Vilhjálmsson, and Hao Yan. Embodiment in conversational interfaces: Rea. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 520–527. ACM, 1999.
106. Marc Cavazza, Fred Charles, and Steven J. Mead. Character-based interactive storytelling. *IEEE Intelligent Systems*, 17(4):17–24, 2002.

107. Marc Cavazza, Fred Charles, and Steven J. Mead. Interacting with virtual characters in interactive storytelling. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems: part 1*, pages 318–325. ACM, 2002.
108. Georgios Chalkiadakis, Edith Elkind, and Michael Wooldridge. Computational aspects of cooperative game theory. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 5(6):1–168, 2011.
109. Alex J. Champandard. *AI game development: Synthetic creatures with learning and reactive behaviors*. New Riders, 2003.
110. Alex J. Champandard. Behavior trees for next-gen game AI. In *Game Developers Conference, Audio Lecture*, 2007.
111. Alex J. Champandard. Understanding Behavior Trees. *AiGameDev.com*, 2007.
112. Alex J. Champandard. Getting started with decision making and control systems. *AI Game Programming Wisdom*, 4:257–264, 2008.
113. Jason C. Chan. Response-order effects in Likert-type scales. *Educational and Psychological Measurement*, 51(3):531–540, 1991.
114. Senthilkumar Chandramohan, Matthieu Geist, Fabrice Lefevre, and Olivier Pietquin. User simulation in dialogue systems using inverse reinforcement learning. In *Interspeech 2011*, pages 1025–1028, 2011.
115. Devendra Singh Chaplot and Guillaume Lample. Arnold: An autonomous agent to play FPS games. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.
116. Darryl Charles and Michaela Black. Dynamic player modelling: A framework for player-centric digital games. In *Proceedings of the International Conference on Computer Games: Artificial Intelligence, Design and Education*, pages 29–35, 2004.
117. Fred Charles, Miguel Lozano, Steven J. Mead, Alicia Fornes Bisquerra, and Marc Cavazza. Planning formalisms and authoring in interactive storytelling. In *Proceedings of TIDSE*, 2003.
118. Guillaume M. J. B. Chaslot, Mark H. M. Winands, H. Jaap van Den Herik, Jos W. H. M. Uiterwijk, and Bruno Bouzy. Progressive strategies for Monte-Carlo tree search. *New Mathematics and Natural Computation*, 4(03):343–357, 2008.
119. Xiang ‘Anthony’ Chen, Tovi Grossman, Daniel J. Wigdor, and George Fitzmaurice. Duet: Exploring joint interactions on a smart phone and a smart watch. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 159–168, 2014.
120. Zhengxing Chen, Magy Seif El-Nasr, Alessandro Canossa, Jeremy Badler, Stefanie Tignor, and Randy Colvin. Modeling individual differences through frequent pattern mining on role-playing game actions. In *Eleventh Artificial Intelligence and Interactive Digital Entertainment Conference, AIIDE*, 2015.
121. Sonia Chernova, Jeff Orkin, and Cynthia Breazeal. Crowdsourcing HRI through online multiplayer games. In *AAAI Fall Symposium: Dialog with Robots*, pages 14–19, 2010.
122. Wei Chu and Zoubin Ghahramani. Preference learning with Gaussian processes. In *Proceedings of the International Conference on Machine learning (ICML)*, pages 137–144, 2005.
123. David Churchill and Michael Buro. Portfolio greedy search and simulation for large-scale combat in StarCraft. In *Computational Intelligence in Games (CIG), 2013 IEEE Conference on*. IEEE, 2013.
124. David Churchill, Mike Preuss, Florian Richoux, Gabriel Synnaeve, Alberto Uriarte, Santiago Ontañón, and Michal Certický. StarCraft Bots and Competitions. In *Encyclopedia of Computer Graphics and Games*. Springer, 2016.
125. Andrea Clerico, Cindy Chamberland, Mark Parent, Pierre-Emmanuel Michon, Sébastien Tremblay, Tiago H. Falk, Jean-Christophe Gagnon, and Philip Jackson. Biometrics and classifier fusion to predict the fun-factor in video gaming. In *IEEE Computational Intelligence and Games Conference*. IEEE, 2016.
126. Carlos A. Coello Coello, Gary B. Lamont, and David A. van Veldhuizen. *Evolutionary algorithms for solving multi-objective problems*. Springer, 2007.
127. Nicholas Cole, Sushil J. Louis, and Chris Miles. Using a genetic algorithm to tune first-person shooter bots. In *Congress on Evolutionary Computation (CEC)*, pages 139–145. IEEE, 2004.

128. Karen Collins. An introduction to procedural music in video games. *Contemporary Music Review*, 28(1):5–15, 2009.
129. Karen Collins. *Playing with sound: a theory of interacting with sound and music in video games*. MIT Press, 2013.
130. Simon Colton. Creativity versus the perception of creativity in computational systems. In *AAAI Spring Symposium: Creative Intelligent Systems*, 2008.
131. Cristina Conati. Intelligent tutoring systems: New challenges and directions. In *IJCAI*, pages 2–7, 2009.
132. Cristina Conati, Abigail Gertner, and Kurt VanLehn. Using Bayesian networks to manage uncertainty in student modeling. *User Modeling and User-Adapted Interaction*, 12(4):371–417, 2002.
133. Cristina Conati and Heather Maclare. Modeling user affect from causes and effects. *User Modeling, Adaptation, and Personalization*, pages 4–15, 2009.
134. John Conway. The game of life. *Scientific American*, 223(4):4, 1970.
135. Michael Cook and Simon Colton. Multi-faceted evolution of simple arcade games. In *IEEE Computational Intelligence and Games*, pages 289–296, 2011.
136. Michael Cook and Simon Colton. Ludus ex machina: Building a 3D game designer that competes alongside humans. In *Proceedings of the 5th International Conference on Computational Creativity*, 2014.
137. Michael Cook, Simon Colton, and Alison Pease. Aesthetic Considerations for Automated Platformer Design. In *AIIDE*, 2012.
138. Seth Cooper, Firas Khatib, Adrien Treuille, Janos Barbero, Jeehyung Lee, Michael Beenen, Andrew Leaver-Fay, David Baker, Zoran Popović, et al. Predicting protein structures with a multiplayer online game. *Nature*, 466(7307):756–760, 2010.
139. Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.
140. Paul T. Costa and Robert R. MacCrae. *Revised NEO personality inventory (NEO PI-R) and NEO five-factor inventory (NEO-FFI): Professional manual*. Psychological Assessment Resources, Incorporated, 1992.
141. Rémi Coulom. Efficient selectivity and backup operators in Monte-Carlo tree search. In *International Conference on Computers and Games*, pages 72–83. Springer, 2006.
142. Rémi Coulom. Computing Elo ratings of move patterns in the game of Go. In *Computer Games Workshop*, 2007.
143. Roddy Cowie and Randolph R. Cornelius. Describing the emotional states that are expressed in speech. *Speech Communication*, 40(1):5–32, 2003.
144. Roddy Cowie, Ellen Douglas-Cowie, Susie Savvidou, Edelle McMahon, Martin Sawey, and Marc Schröder. ‘FEELTRACE’: An instrument for recording perceived emotion in real time. In *ISCA Tutorial and Research Workshop (ITRW) on Speech and Emotion*, 2000.
145. Roddy Cowie and Martin Sawey. GTrace-General trace program from Queen’s University, Belfast, 2011.
146. Peter I. Cowling, Edward J. Powley, and Daniel Whitehouse. Information set Monte Carlo tree search. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(2):120–143, 2012.
147. Koby Crammer and Yoram Singer. Pranking with ranking. *Advances in Neural Information Processing Systems*, 14:641–647, 2002.
148. Chris Crawford. *Chris Crawford on interactive storytelling*. New Riders, 2012.
149. Mihaly Csikszentmihalyi. *Creativity: Flow and the psychology of discovery and invention*. New York: Harper Collins, 1996.
150. Mihaly Csikszentmihalyi. *Beyond boredom and anxiety*. Jossey-Bass, 2000.
151. Mihaly Csikszentmihalyi. *Toward a psychology of optimal experience*. Springer, 2014.
152. George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, 1989.
153. Ryan S. J. d. Baker, Gregory R. Moore, Angela Z. Wagner, Jessica Kalka, Aatish Salvi, Michael Karabinos, Colin A. Ashe, and David Yaron. The Dynamics between Student Affect and Behavior Occurring Outside of Educational Software. In *Affective Computing and Intelligent Interaction*, pages 14–24. Springer, 2011.

154. Anders Dahlbom and Lars Niklasson. Goal-Directed Hierarchical Dynamic Scripting for RTS Games. In *AIIDE*, pages 21–28, 2006.
155. Steve Dahlskog and Julian Togelius. Patterns as objectives for level generation. In *Proceedings of the International Conference on the Foundations of Digital Games*. ACM, 2013.
156. Steve Dahlskog, Julian Togelius, and Mark J. Nelson. Linear levels through n-grams. In *Proceedings of the 18th International Academic MindTrek Conference: Media Business, Management, Content & Services*, pages 200–206. ACM, 2014.
157. Antonio R. Damasio, Barry J. Everitt, and Dorothy Bishop. The somatic marker hypothesis and the possible functions of the prefrontal cortex [and discussion]. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 351(1346):1413–1420, 1996.
158. Gustavo Danzi, Andrade Hugo Pimentel Santana, André Wilson Brotto Furtado, André Roberto Gouveia, Amaral Leitao, and Geber Lisboa Ramalho. Online adaptation of computer games agents: A reinforcement learning approach. In *II Workshop de Jogos e Entretenimento Digital*, pages 105–112, 2003.
159. Isaac M. Dart, Gabriele De Rossi, and Julian Togelius. SpeedRock: procedural rocks through grammars and evolution. In *Proceedings of the 2nd International Workshop on Procedural Content Generation in Games*. ACM, 2011.
160. Fernando de Mesentier Silva, Scott Lee, Julian Togelius, and Andy Nealen. AI-based Playtesting of Contemporary Board Games. In *Proceedings of Foundations of Digital Games (FDG)*, 2017.
161. Maarten de Waard, Diederik M. Roijers, and Sander Bakkes. Monte Carlo tree search with options for general video game playing. In *Computational Intelligence and Games (CIG), 2016 IEEE Conference on*. IEEE, 2016.
162. Edward L. Deci and Richard M. Ryan. *Intrinsic motivation*. Wiley Online Library, 1975.
163. Erik D. Demaine, Giovanni Viglietta, and Aaron Williams. Super Mario Bros. is Harder/Easier than We Thought. In *Proceedings of the 8th International Conference on Fun with Algorithms (FUN 2016)*, pages 13:1–13:14, La Maddalena, Italy, June 8–10 2016.
164. Jack Dennerlein, Theodore Becker, Peter Johnson, Carson Reynolds, and Rosalind W. Picard. Frustrating computer users increases exposure to physical factors. In *Proceedings of the International Ergonomics Association (IEA)*, 2003.
165. Jörg Denzinger, Kevin Loose, Darryl Gates, and John W. Buchanan. Dealing with Parameterized Actions in Behavior Testing of Commercial Computer Games. In *IEEE Symposium on Computational Intelligence and Games*, 2005.
166. L. Devillers, R. Cowie, J. C. Martin, E. Douglas-Cowie, S. Abrilian, and M. McRorie. Real life emotions in French and English TV video clips: an integrated annotation protocol combining continuous and discrete approaches. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC 2006)*, Genoa, Italy, page 22, 2006.
167. Ravi Dhar and Itamar Simonson. The effect of forced choice on choice. *Journal of Marketing Research*, 40(2), 2003.
168. Joao Dias, Samuel Mascarenhas, and Ana Paiva. Fatima modular: Towards an agent architecture with a generic appraisal framework. In *Emotion Modeling*, pages 44–56. Springer, 2014.
169. Kevin Dill. A pattern-based approach to modular AI for Games. *Game Programming Gems*, 8:232–243, 2010.
170. Kevin Dill. Introducing GAIA: A Reusable, Extensible architecture for AI behavior. In *Proceedings of the 2012 Spring Simulation Interoperability Workshop*, 2012.
171. Kevin Dill and L. Martin. A game AI approach to autonomous control of virtual characters. In *Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC)*, 2011.
172. Sidney D’Mello and Art Graesser. Automatic detection of learner’s affect from gross body language. *Applied Artificial Intelligence*, 23(2):123–150, 2009.
173. Joris Dormans. Adventures in level design: generating missions and spaces for action adventure games. In *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*. ACM, 2010.

174. Joris Dormans and Sander Bakkes. Generating missions and spaces for adaptable play experiences. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(3):216–228, 2011.
175. Anders Drachen, Lennart Nacke, Georgios N. Yannakakis, and Anja Lee Pedersen. Correlation between heart rate, electrodermal activity and player experience in first-person shooter games. In *Proceedings of the SIGGRAPH Symposium on Video Games*. ACM-SIGGRAPH Publishers, 2010.
176. Anders Drachen, Alessandro Canossa, and Georgios N. Yannakakis. Player modeling using self-organization in Tomb Raider: Underworld. In *Proceedings of the 2009 IEEE Symposium on Computational Intelligence and Games*, pages 1–8. IEEE, 2009.
177. Anders Drachen and Matthias Schubert. Spatial game analytics. In *Game Analytics*, pages 365–402. Springer, 2013.
178. Anders Drachen, Christian Thurau, Julian Togelius, Georgios N. Yannakakis, and Christian Bauckhage. Game Data Mining. In *Game Analytics*, pages 205–253. Springer, 2013.
179. H. Drucker, C.J. C. Burges, L. Kaufman, A. Smola, and V. Vapnik. Support vector regression machines. In *Advances in Neural Information Processing Systems (NIPS)*, pages 155–161. Morgan Kaufmann Publishers, 1997.
180. David S. Ebert. *Texturing & modeling: a procedural approach*. Morgan Kaufmann, 2003.
181. Marc Ebner, John Levine, Simon M. Lucas, Tom Schaul, Tommy Thompson, and Julian Togelius. Towards a video game description language. *Dagstuhl Follow-Ups*, 6, 2013.
182. Arthur S. Eddington. The Constants of Nature. In *The World of Mathematics 2*, pages 1074–1093. Simon & Schuster, 1956.
183. Arjan Eggels, Sumedha Kshirsagar, and Nadia Magnenat-Thalmann. Generic personality and emotion simulation for conversational agents. *Computer animation and virtual worlds*, 15(1):1–13, 2004.
184. Agoston E. Eiben and James E. Smith. *Introduction to Evolutionary Computing*. Springer, 2003.
185. Magy Seif El-Nasr. Intelligent lighting for game environments. *Journal of Game Development*, 2005.
186. Magy Seif El-Nasr, Anders Drachen, and Alessandro Canossa. *Game analytics: Maximizing the value of player data*. Springer, 2013.
187. Magy Seif El-Nasr, Shree Durga, Mariya Shiyko, and Carmen Sceppa. Data-driven retrospective interviewing (DDRI): a proposed methodology for formative evaluation of pervasive games. *Entertainment Computing*, 11:1–19, 2015.
188. Magy Seif El-Nasr, Athanasios Vasilakos, Chinmay Rao, and Joseph Zupko. Dynamic intelligent lighting for directing visual attention in interactive 3-D scenes. *Computational Intelligence and AI in Games, IEEE Transactions on*, 1(2):145–153, 2009.
189. Magy Seif El-Nasr, John Yen, and Thomas R. Ioerger. Flame—fuzzy logic adaptive model of emotions. *Autonomous Agents and Multi-Agent Systems*, 3(3):219–257, 2000.
190. Mirjam Palosaari Eladhari and Michael Mateas. Semi-autonomous avatars in World of Minds: A case study of AI-based game design. In *Proceedings of the 2008 International Conference on Advances in Computer Entertainment Technology*, pages 201–208. ACM, 2008.
191. Mirjam Palosaari Eladhari and Michael Sellers. Good moods: outlook, affect and mood in dynemotion and the mind module. In *Proceedings of the 2008 Conference on Future Play: Research, Play, Share*, pages 1–8. ACM, 2008.
192. George Skaff Elias, Richard Garfield, K. Robert Gutschera, and Peter Whitley. *Characteristics of games*. MIT Press, 2012.
193. David K. Elson and Mark O. Riedl. A lightweight intelligent virtual cinematography system for machinima production. In *AIIDE*, pages 8–13, 2007.
194. Nathan Ensmenger. Is Chess the Drosophila of AI? A Social History of an Algorithm. *Social Studies of Science*, 42(1):5–30, 2012.
195. Ido Erev and Alvin E. Roth. Predicting how people play games: Reinforcement learning in experimental games with unique, mixed strategy equilibria. *American Economic Review*, pages 848–881, 1998.

196. Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 226–231, 1996.
197. Richard Evans and Emily Short. Versu—a simulationist storytelling system. *IEEE Transactions on Computational Intelligence and AI in Games*, 6(2):113–130, 2014.
198. Vincent E. Farrugia, Héctor P. Martínez, and Georgios N. Yannakakis. The preference learning toolbox. *arXiv preprint arXiv:1506.01709*, 2015.
199. Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. *arXiv preprint arXiv:1708.00524*, 2017.
200. Lisa A. Feldman. Valence focus and arousal focus: Individual differences in the structure of affective experience. *Journal of personality and social psychology*, 69(1):153, 1995.
201. David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A. Kalyanpur, Adam Lally, J. William Murdock, Eric Nyberg, John Prager, Nico Schlaefer, and Chris Welty. Building Watson: An overview of the DeepQA project. *AI Magazine*, 31(3):59–79, 2010.
202. Hilmar Finnsson and Yngvi Björnsson. Learning simulation control in general game-playing agents. In *AAAI*, pages 954–959, 2010.
203. Jacob Fischer, Nikolaj Falsted, Mathias Vielwerth, Julian Togelius, and Sebastian Risi. Monte-Carlo Tree Search for Simulated Car Racing. In *Proceedings of FDG*, 2015.
204. John H. Flavell. *The developmental psychology of Jean Piaget*. Ardent Media, 1963.
205. Dario Floreano, Peter Dürr, and Claudio Mattiussi. Neuroevolution: from architectures to learning. *Evolutionary Intelligence*, 1(1):47–62, 2008.
206. Dario Floreano, Toshifumi Kato, Davide Marocco, and Eric Sauser. Coevolution of active vision and feature selection. *Biological Cybernetics*, 90(3):218–228, 2004.
207. David B. Fogel. *Blondie24: Playing at the Edge of AI*. Morgan Kaufmann, 2001.
208. David B. Fogel, Timothy J. Hays, Sarah L. Hahn, and James Quon. The Blondie25 chess program competes against Fritz 8.0 and a human chess master. In *Computational Intelligence and Games, 2006 IEEE Symposium on*, pages 230–235. IEEE, 2006.
209. Tom Forsyth. Cellular automata for physical modelling. *Game Programming Gems*, 3:200–214, 2002.
210. Alain Fournier, Don Fussell, and Loren Carpenter. Computer rendering of stochastic models. *Communications of the ACM*, 25(6):371–384, 1982.
211. Michael Freed, Travis Bear, Herrick Goldman, Geoffrey Hyatt, Paul Reber, A. Sylvan, and Joshua Tauber. Towards more human-like computer opponents. In *Working Notes of the AAAI Spring Symposium on Artificial Intelligence and Interactive Entertainment*, pages 22–26, 2000.
212. Nico Frijda. *The Emotions*. Cambridge University Press, Englewood Cliffs, NJ, 1986.
213. Frederik Frydenberg, Kasper R. Andersen, Sebastian Risi, and Julian Togelius. Investigating MCTS modifications in general video game playing. In *Computational Intelligence and Games (CIG), 2015 IEEE Conference on*, pages 107–113. IEEE, 2015.
214. Drew Fudenberg and David K. Levine. *The theory of learning in games*. MIT Press, 1998.
215. J. Fürnkranz and E. Hüllermeier. *Preference learning*. Springer, 2010.
216. Raluca D. Gaina, Jialin Liu, Simon M. Lucas, and Diego Pérez-Liébana. Analysis of Vanilla Rolling Horizon Evolution Parameters in General Video Game Playing. In *European Conference on the Applications of Evolutionary Computation*, pages 418–434. Springer, 2017.
217. Maurizio Garbarino, Simone Tognetti, Matteo Matteucci, and Andrea Bonarini. Learning general preference models from physiological responses in video games: How complex is it? In *Affective Computing and Intelligent Interaction*, pages 517–526. Springer, 2011.
218. Pablo García-Sánchez, Alberto Tonda, Giovanni Squillero, Antonio Mora, and Juan J. Merelo. Evolutionary deckbuilding in Hearthstone. In *Computational Intelligence and Games (CIG), 2016 IEEE Conference on*. IEEE, 2016.
219. Tom A. Garner. From Sinewaves to Physiologically-Adaptive Soundscapes: The Evolving Relationship Between Sound and Emotion in Video Games. In *Emotion in Games: Theory and Praxis*, pages 197–214. Springer, 2016.

220. Tom A. Garner and Mark Grimshaw. Sonic virtuality: Understanding audio in a virtual world. *The Oxford Handbook of Virtuality*, 2014.
221. H. P. Gasselseder. Re-scoring the games score: Dynamic music and immersion in the ludonarrative. In *Proceedings of the Intelligent Human Computer Interaction conference*, 2014.
222. Jakub Gemrot, Rudolf Kadlec, Michal Bída, Ondřej Burkert, Radek Píbil, Jan Havlíček, Lukáš Zemčák, Juraj Šimlovič, Radim Vansa, Michal Štolba, Tomáš Plch, and Cyril Brom. Pogamut 3 can assist developers in building AI (not only) for their videogame agents. In *Agents for games and simulations*, pages 1–15. Springer, 2009.
223. Michael Genesereth, Nathaniel Love, and Barney Pell. General game playing: Overview of the AAAI competition. *AI Magazine*, 26(2):62, 2005.
224. Michael Georgeff, Barney Pell, Martha Pollack, Milind Tambe, and Michael Wooldridge. The belief-desire-intention model of agency. In *International Workshop on Agent Theories, Architectures, and Languages*, pages 1–10. Springer, 1998.
225. Kallirroi Georgila, James Henderson, and Oliver Lemon. Learning user simulations for information state update dialogue systems. In *Interspeech*, pages 893–896, 2005.
226. Panayiotis G. Georgiou, Matthew P. Black, Adam C. Lammert, Brian R. Baucom, and Shrikanth S. Narayanan. “That’s Aggravating, Very Aggravating”: Is It Possible to Classify Behaviors in Couple Interactions Using Automatically Derived Lexical Features? In *Affective Computing and Intelligent Interaction*, pages 87–96. Springer, 2011.
227. Maryrose Gerardi, Barbara Olasov Rothbaum, Kerry Ressler, Mary Heekin, and Albert Rizzo. Virtual reality exposure therapy using a virtual Iraq: case report. *Journal of Traumatic Stress*, 21(2):209–213, 2008.
228. Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated Planning: theory and practice*. Elsevier, 2004.
229. Spyridon Giannatos, Yun-Gyung Cheong, Mark J. Nelson, and Georgios N. Yannakakis. Generating narrative action schemas for suspense. In *Eighth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2012.
230. Arthur Gill. *Introduction to the theory of Finite-State Machines*. McGraw-Hill, 1962.
231. Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.
232. Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.
233. Nitesh Goyal, Gilly Leshed, Dan Cosley, and Susan R. Fussell. Effects of implicit sharing in collaborative analysis. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 129–138, 2014.
234. Katja Grace, John Salvatier, Allan Dafoe, Baobao Zhang, and Owain Evans. When Will AI Exceed Human Performance? Evidence from AI Experts. *arXiv preprint arXiv:1705.08807*, 2017.
235. Thore Graepel, Ralf Herbrich, and Julian Gold. Learning to fight. In *Proceedings of the International Conference on Computer Games: Artificial Intelligence, Design and Education*, pages 193–200, 2004.
236. Joseph F. Grafsgaard, Kristy Elizabeth Boyer, and James C. Lester. Predicting facial indicators of confusion with hidden Markov models. In *Proceedings of International Conference on Affective Computing and Intelligent Interaction (ACII)*, pages 97–106. Springer, 2011.
237. Jonathan Gratch. Emile: Marshalling passions in training and education. In *Proceedings of the Fourth International Conference on Autonomous Agents*, pages 325–332. ACM, 2000.
238. Jonathan Gratch and Stacy Marsella. A domain-independent framework for modeling emotion. *Cognitive Systems Research*, 5(4):269–306, 2004.
239. Jonathan Gratch and Stacy Marsella. Evaluating a computational model of emotion. *Autonomous Agents and Multi-Agent Systems*, 11(1):23–43, 2005.
240. Daniele Gravina, Antonios Liapis, and Georgios N. Yannakakis. Constrained surprise search for content generation. In *Computational Intelligence and Games (CIG), 2016 IEEE Conference on*. IEEE, 2016.

241. Elin Rut Gudnadottir, Alaina K. Jensen, Yun-Gyung Cheong, Julian Togelius, Byung Chull Bae, and Christoffer Holmgård Pedersen. Detecting predatory behaviour in online game chats. In *The 2nd Workshop on Games and NLP*, 2014.
242. Johan Hagelbck. Potential-field based navigation in StarCraft. In *IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 2012.
243. Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The WEKA data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18, 2009.
244. Jiawei Han and Micheline Kamber. *Data mining: concepts and techniques*. Morgan Kaufmann, 2006.
245. Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.
246. Daniel Damir Harabor and Alban Grastien. Online Graph Pruning for Pathfinding on Grid Maps. In *AAAI*, 2011.
247. Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. Correction to a formal basis for the heuristic determination of minimum cost paths. *ACM SIGART Bulletin*, (37):28–29, 1972.
248. Ken Hartsook, Alexander Zook, Sauvik Das, and Mark O. Riedl. Toward supporting stories with procedurally generated game worlds. In *Computational Intelligence and Games (CIG), 2011 IEEE Conference on*, pages 297–304. IEEE, 2011.
249. Erin J. Hastings, Ratan K. Guha, and Kenneth O. Stanley. Automatic content generation in the Galactic Arms Race video game. *IEEE Transactions on Computational Intelligence and AI in Games*, 1(4):245–263, 2009.
250. Erin J. Hastings, Ratan K. Guha, and Kenneth O. Stanley. Evolving content in the Galactic Arms Race video game. In *IEEE Symposium on Computational Intelligence and Games*, pages 241–248. IEEE, 2009.
251. Matthew Hausknecht, Joel Lehman, Risto Miikkulainen, and Peter Stone. A neuroevolution approach to general Atari game playing. *IEEE Transactions on Computational Intelligence and AI in Games*, 6(4):355–366, 2014.
252. Brian Hawkins. *Real-Time Cinematography for Games (Game Development Series)*. Charles River Media, Inc., 2004.
253. Simon Haykin. *Neural Networks: A Comprehensive Foundation*. Macmillan College Publishing Company Inc., Upper Saddle River, NJ, USA, 1998.
254. Richard L. Hazlett. Measuring emotional valence during interactive experiences: boys at video game play. In *Proceedings of SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pages 1023–1026. ACM, 2006.
255. Jennifer Healey. Recording affect in the field: Towards methods and metrics for improving ground truth labels. In *Affective Computing and Intelligent Interaction*, pages 107–116. Springer, 2011.
256. D. O. Hebb. *The Organization of Behavior*. Wiley, New York, 1949.
257. Norbert Heijne and Sander Bakkes. Procedural Zelda: A PCG Environment for Player Experience Research. In *Proceedings of the International Conference on the Foundations of Digital Games*. ACM, 2017.
258. Harry Helson. *Adaptation-level theory*. Harper & Row, 1964.
259. Ralf Herbrich, Michael E. Tipping, and Mark Hatton. Personalized behavior of computer controlled avatars in a virtual reality environment, August 15 2006. US Patent 7,090,576.
260. Javier Hernandez, Rob R. Morris, and Rosalind W. Picard. Call center stress recognition with person-specific models. In *Affective Computing and Intelligent Interaction*, pages 125–134. Springer, 2011.
261. David Hilbert. Über die stetige Abbildung einer Linie auf ein Flächenstück. *Mathematische Annalen*, 38(3):459–460, 1891.
262. Philip Hingston. A Turing test for computer game bots. *IEEE Transactions on Computational Intelligence and AI in Games*, 1(3):169–186, 2009.
263. Philip Hingston. A new design for a Turing test for bots. In *Computational Intelligence and Games (CIG), 2010 IEEE Symposium on*, pages 345–350. IEEE, 2010.

264. Philip Hingston. *Believable Bots: Can Computers Play Like People?* Springer, 2012.
265. Philip Hingston, Clare Bates Congdon, and Graham Kendall. Mobile games with intelligence: A killer application? In *Computational Intelligence in Games (CIG), 2013 IEEE Conference on*, pages 1–7. IEEE, 2013.
266. Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
267. Christoffer Holmgård, Antonios Liapis, Julian Togelius, and Georgios N. Yannakakis. Evolving personas for player decision modeling. In *Computational Intelligence and Games (CIG), 2014 IEEE Conference on*. IEEE, 2014.
268. Christoffer Holmgård, Antonios Liapis, Julian Togelius, and Georgios N. Yannakakis. Generative agents for player decision modeling in games. In *FDG*, 2014.
269. Christoffer Holmgård, Antonios Liapis, Julian Togelius, and Georgios N. Yannakakis. Personas versus clones for player decision modeling. In *International Conference on Entertainment Computing*, pages 159–166. Springer, 2014.
270. Christoffer Holmgård, Georgios N. Yannakakis, Karen-Inge Karstoft, and Henrik Steen Andersen. Stress detection for PTSD via the Startlemart game. In *Affective Computing and Intelligent Interaction (ACII), 2013 Humaine Association Conference on*, pages 523–528. IEEE, 2013.
271. Christoffer Holmgård, Georgios N. Yannakakis, Héctor P. Martínez, and Karen-Inge Karstoft. To rank or to classify? Annotating stress for reliable PTSD profiling. In *Affective Computing and Intelligent Interaction (ACII), 2015 International Conference on*, pages 719–725. IEEE, 2015.
272. Christoffer Holmgård, Georgios N. Yannakakis, Héctor P. Martínez, Karen-Inge Karstoft, and Henrik Steen Andersen. Multimodal PTSD characterization via the Startlemart game. *Journal on Multimodal User Interfaces*, 9(1):3–15, 2015.
273. Nils Iver Holtar, Mark J. Nelson, and Julian Togelius. Audiooverdrive: Exploring bidirectional communication between music and gameplay. In *Proceedings of the 2013 International Computer Music Conference*, pages 124–131, 2013.
274. Vincent Hom and Joe Marks. Automatic design of balanced board games. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AI-IDE)*, pages 25–30, 2007.
275. Amy K. Hoover, William Cachia, Antonios Liapis, and Georgios N. Yannakakis. AudioInSpace: Exploring the Creative Fusion of Generative Audio, Visuals and Gameplay. In *Evolutionary and Biologically Inspired Music, Sound, Art and Design*, pages 101–112. Springer, 2015.
276. Amy K. Hoover, Paul A. Szerlip, and Kenneth O. Stanley. Functional scaffolding for composing additional musical voices. *Computer Music Journal*, 2014.
277. Amy K. Hoover, Julian Togelius, and Georgios N. Yannakakis. Composing video game levels with music metaphors through functional scaffolding. In *First Computational Creativity and Games Workshop, ICCC*, 2015.
278. John J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, 1982.
279. Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.
280. Ben Houge. Cell-based music organization in Tom Clancy’s EndWar. In *Demo at the AIIDE 2012 Workshop on Musical Metacreation*, 2012.
281. Ryan Houlette. *Player Modeling for Adaptive Games. AI Game Programming Wisdom II*, pages 557–566. Charles River Media, Inc., 2004.
282. Andrew Howlett, Simon Colton, and Cameron Browne. Evolving pixel shaders for the prototype video game Subversion. In *The Thirty Sixth Annual Convention of the Society for the Study of Artificial Intelligence and Simulation of Behaviour (AISB10), De Montfort University, Leicester, UK, 30th March*, 2010.
283. Johanna Höytsniemi, Perttu Hämäläinen, Laura Turkki, and Teppo Rouvi. Children’s intuitive gestures in vision-based action games. *Communications of the ACM*, 48(1):44–50, 2005.

284. Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks*, 13(2):415–425, 2002.
285. Feng-Hsiung Hsu. *Behind Deep Blue: Building the computer that defeated the world chess champion*. Princeton University Press, 2002.
286. Wijnand IJsselsteijn, Karolien Poels, and Y. A. W. De Kort. The game experience questionnaire: Development of a self-report measure to assess player experiences of digital games. *TU Eindhoven, Eindhoven, The Netherlands*, 2008.
287. Interactive Data Visualization. SpeedTree, 2010. <http://www.speedtree.com/>.
288. Aaron Isaksen, Dan Gopstein, Julian Togelius, and Andy Nealen. Discovering unique game variants. In *Computational Creativity and Games Workshop at the 2015 International Conference on Computational Creativity*, 2015.
289. Aaron Isaksen, Daniel Gopstein, and Andrew Nealen. Exploring Game Space Using Survival Analysis. In *Proceedings of Foundations of Digital Games (FDG)*, 2015.
290. Katherine Isbister and Noah Schaffer. *Game usability: Advancing the player experience*. CRC Press, 2015.
291. Damian Isla. Handling complexity in the Halo 2 AI. In *Game Developers Conference*, 2005.
292. Damian Isla and Bruce Blumberg. New challenges for character-based AI for games. In *Proceedings of the AAAI Spring Symposium on AI and Interactive Entertainment*, pages 41–45. AAAI Press, 2002.
293. Susan A. Jackson and Robert C. Eklund. Assessing flow in physical activity: the flow state scale-2 and dispositional flow scale-2. *Journal of Sport & Exercise Psychology*, 24(2), 2002.
294. Emil Juul Jacobsen, Rasmus Greve, and Julian Togelius. Monte Mario: platfroming with MCTS. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, pages 293–300. ACM, 2014.
295. Alexander Jaffe, Alex Miller, Erik Andersen, Yun-En Liu, Anna Karlin, and Zoran Popovic. Evaluating competitive game balance with restricted play. In *AIIDE*, 2012.
296. Rishabh Jain, Aaron Isaksen, Christoffer Holmgård, and Julian Togelius. Autoencoders for level generation, repair, and recognition. In *ICCC Workshop on Computational Creativity and Games*, 2016.
297. Daniel Jallov, Sebastian Risi, and Julian Togelius. EvoCommander: A Novel Game Based on Evolving and Switching Between Artificial Brains. *IEEE Transactions on Computational Intelligence and AI in Games*, 9(2):181–191, 2017.
298. Susan Jamieson. Likert scales: how to (ab) use them. *Medical Education*, 38(12):1217–1218, 2004.
299. Aki Järvinen. Gran stylissimo: The audiovisual elements and styles in computer and video games. In *Proceedings of Computer Games and Digital Cultures Conference*, 2002.
300. Arnav Jhala and R. Michael Young. Cinematic visual discourse: Representation, generation, and evaluation. *Computational Intelligence and AI in Games, IEEE Transactions on*, 2(2):69–81, 2010.
301. Yuu Jinnai and Alex S. Fukunaga. Learning to prune dominated action sequences in online black-box planning. In *AAAI*, pages 839–845, 2017.
302. Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. *Machine Learning: ECML-98*, pages 137–142, 1998.
303. Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery in Data Mining (KDD)*, pages 133–142. ACM, 2002.
304. Lawrence Johnson, Georgios N. Yannakakis, and Julian Togelius. Cellular automata for real-time generation of infinite cave levels. In *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*. ACM, 2010.
305. Matthew Johnson, Katja Hofmann, Tim Hutton, and David Bignell. The Malmo Platform for Artificial Intelligence Experimentation. In *IJCAI*, pages 4246–4247, 2016.
306. Tom Johnstone and Klaus R. Scherer. Vocal communication of emotion. In *Handbook of emotions*, pages 220–235. Guilford Press, New York, 2000.

307. German Gutierrez Jorge Munoz and Araceli Sanchis. Towards imitation of human driving style in car racing games. In Philip Hingston, editor, *Believable Bots: Can Computers Play Like People?* Springer, 2012.
308. Patrik N. Juslin and Klaus R. Scherer. *Vocal expression of affect*. Oxford University Press, Oxford, UK, 2005.
309. Niels Justesen, Tobias Mahlmann, and Julian Togelius. Online evolution for multi-action adversarial games. In *European Conference on the Applications of Evolutionary Computation*, pages 590–603. Springer, 2016.
310. Niels Justesen and Sebastian Risi. Continual Online Evolutionary Planning for In-Game Build Order Adaptation in StarCraft. In *Proceedings of the Conference on Genetic and Evolutionary Computation (GECCO)*, 2017.
311. Niels Justesen, Bálint Tillman, Julian Togelius, and Sebastian Risi. Script-and cluster-based UCT for StarCraft. In *Computational Intelligence and Games (CIG), 2014 IEEE Conference on*. IEEE, 2014.
312. Tróndur Justinussen, Peter Hald Rasmussen, Alessandro Canossa, and Julian Togelius. Resource systems in games: An analytical approach. In *Computational Intelligence and Games (CIG), 2012 IEEE Conference on*, pages 171–178. IEEE, 2012.
313. Jesper Juul. Games telling stories. *Game Studies*, 1(1):45, 2001.
314. Jesper Juul. *A casual revolution: Reinventing video games and their players*. MIT Press, 2010.
315. Souhila Kaci. *Working with preferences: Less is more*. Springer, 2011.
316. Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.
317. Daniel Kahneman. A perspective on judgment and choice: mapping bounded rationality. *American psychologist*, 58(9):697, 2003.
318. Daniel Kahneman and Jason Riis. Living, and thinking about it: Two perspectives on life. *The science of well-being*, pages 285–304, 2005.
319. Theofanis Kannetis and Alexandros Potamianos. Towards adapting fantasy, curiosity and challenge in multimodal dialogue systems for preschoolers. In *Proceedings of International Conference on Multimodal Interfaces (ICMI)*, pages 39–46. ACM, 2009.
320. Theofanis Kannetis, Alexandros Potamianos, and Georgios N. Yannakakis. Fantasy, curiosity and challenge as adaptation indicators in multimodal dialogue systems for preschoolers. In *Proceedings of the 2nd Workshop on Child, Computer and Interaction*. ACM, 2009.
321. Ashish Kapoor, Winslow Burleson, and Rosalind W. Picard. Automatic prediction of frustration. *International Journal of Human-Computer Studies*, 65(8):724–736, 2007.
322. Sergey Karakovskiy and Julian Togelius. The Mario AI benchmark and competitions. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):55–67, 2012.
323. Daniël Karavolos, Anders Bouwer, and Rafael Bidarra. Mixed-initiative design of game levels: Integrating mission and space into level generation. In *Proceedings of the 10th International Conference on the Foundations of Digital Games*, 2015.
324. Daniel Karavolos, Antonios Liapis, and Georgios N. Yannakakis. Learning the patterns of balance in a multi-player shooter game. In *Proceedings of the FDG workshop on Procedural Content Generation in Games*, 2017.
325. Kostas Karpouzis and Georgios N. Yannakakis. *Emotion in Games: Theory and Praxis*. Springer, 2016.
326. Kostas Karpouzis, Georgios N. Yannakakis, Noor Shaker, and Stylianos Asteriadis. The Platformer Experience Dataset. In *Affective Computing and Intelligent Interaction (ACII), 2015 International Conference on*, pages 712–718. IEEE, 2015.
327. Igor V. Karpov, Leif Johnson, and Risto Miikkulainen. Evaluation methods for active human-guided neuroevolution in games. In *2012 AAAI Fall Symposium on Robots Learning Interactively from Human Teachers (RLIHT)*, 2012.
328. Igor V. Karpov, Jacob Schrum, and Risto Miikkulainen. Believable bot navigation via playback of human traces. In Philip Hingston, editor, *Believable Bots: Can Computers Play Like People?* Springer, 2012.

329. Leonard Kaufman and Peter J. Rousseeuw. *Clustering by means of medoids*. North-Holland, 1987.
330. Leonard Kaufman and Peter J. Rousseeuw. *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons, 2009.
331. Richard Kaye. Minesweeper is NP-complete. *The Mathematical Intelligencer*, 22(2):9–15, 2000.
332. Markus Kemmerling and Mike Preuss. Automatic adaptation to generated content via car setup optimization in TORCS. In *Computational Intelligence and Games (CIG), 2010 IEEE Symposium on*, pages 131–138. IEEE, 2010.
333. Michał Kempka, Marek Wydmuch, Grzegorz Runc, Jakub Toczek, and Wojciech Jaśkowski. Vizdoom: A doom-based AI research platform for visual reinforcement learning. *arXiv preprint arXiv:1605.02097*, 2016.
334. Graham Kendall, Andrew J. Parkes, and Kristian Spoerer. A Survey of NP-Complete Puzzles. *ICGA Journal*, 31(1):13–34, 2008.
335. Manuel Kerssemakers, Jeppe Tuxen, Julian Togelius, and Georgios N. Yannakakis. A procedural procedural level generator generator. In *Computational Intelligence and Games (CIG), 2012 IEEE Conference on*, pages 335–341. IEEE, 2012.
336. Rilla Khaled and Georgios N. Yannakakis. Village voices: An adaptive game for conflict resolution. In *Proceedings of FDG*, pages 425–426, 2013.
337. Ahmed Khalifa, Aaron Isaksen, Julian Togelius, and Andy Nealen. Modifying MCTS for Human-like General Video Game Playing. In *Proceedings of IJCAI*, 2016.
338. Ahmed Khalifa, Diego Perez-Liebana, Simon M. Lucas, and Julian Togelius. General video game level generation. In *Proceedings of IJCAI*, 2016.
339. K-J Kim, Heejin Choi, and Sung-Bae Cho. Hybrid of evolution and reinforcement learning for Othello players. In *Computational Intelligence and Games, 2007. CIG 2007. IEEE Symposium on*, pages 203–209. IEEE, 2007.
340. Kyung-Min Kim, Chang-Jun Nan, Jung-Woo Ha, Yu-Jung Heo, and Byoung-Tak Zhang. Pororobot: A deep learning robot that plays video Q&A games. In *AAAI 2015 Fall Symposium on AI for Human-Robot Interaction (AI-HRI 2015)*, 2015.
341. Steven Orla Kimbrough, Gary J. Koehler, Ming Lu, and David Harlan Wood. On a Feasible-Infeasible Two-Population (FI-2Pop) genetic algorithm for constrained optimization: Distance tracing and no free lunch. *European Journal of Operational Research*, 190(2):310–327, 2008.
342. Diederik P. Kingma and Max Welling. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.
343. A. Kleinsmith and N. Bianchi-Berthouze. Affective body expression perception and recognition: A survey. *IEEE Transactions on Affective Computing*, 2012.
344. Andrea Kleinsmith and Nadia Bianchi-Berthouze. Form as a cue in the automatic recognition of non-acted affective body expressions. In *Affective Computing and Intelligent Interaction*, pages 155–164. Springer, 2011.
345. Yana Knight, Héctor Pérez Martínez, and Georgios N. Yannakakis. Space maze: Experience-driven game camera control. In *FDG*, pages 427–428, 2013.
346. Matthias J. Koeppl, Roger N. Gunn, Andrew D. Lawrence, Vincent J. Cunningham, Alain Dagher, Tasmin Jones, David J. Brooks, C. J. Bench, and P. M. Grasby. Evidence for striatal dopamine release during a video game. *Nature*, 393(6682):266–268, 1998.
347. Teuvo Kohonen. *Self-Organizing Maps*. Springer, Secaucus, NJ, USA, 3rd edition, 2001.
348. Andrey N. Kolmogorov. On the representation of continuous functions of several variables by superposition of continuous functions of one variable and addition. *Russian, American Mathematical Society Translation 28 (1963) 55-59. Doklady Akademii Nauk SSR*, 14(5):953–956, 1957.
349. Richard Konečný. Modeling of fighting game players. Master’s thesis, Institute of Digital Games, University of Malta, 2016.
350. Michael Kosfeld, Markus Heinrichs, Paul J. Zak, Urs Fischbacher, and Ernst Fehr. Oxytocin increases trust in humans. *Nature*, 435(7042):673–676, 2005.

351. Raph Koster. *Theory of fun for game design*. O'Reilly Media, Inc., 2013.
352. Bartosz Kostka, Jaroslaw Kwiecien, Jakub Kowalski, and Paweł Rychlikowski. Text-based Adventures of the Golovin AI Agent. *arXiv preprint arXiv:1705.05637*, 2017.
353. Jan Koutník, Giuseppe Cuccu, Jürgen Schmidhuber, and Faustino Gomez. Evolving large-scale neural networks for vision-based reinforcement learning. In *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*, pages 1061–1068. ACM, 2013.
354. Jakub Kowalski and Andrzej Kisielewicz. Towards a Real-time Game Description Language. In *ICAART (2)*, pages 494–499, 2016.
355. Jakub Kowalski and Marek Szykula. Evolving chess-like games using relative algorithm performance profiles. In *European Conference on the Applications of Evolutionary Computation*, pages 574–589. Springer, 2016.
356. John R. Koza. *Genetic programming: on the programming of computers by means of natural selection*. MIT Press, 1992.
357. Teofebano Kristo and Nur Ulfa Maulidevi. Deduction of fighting game countermeasures using Neuroevolution of Augmenting Topologies. In *Data and Software Engineering (ICoDSE), 2016 International Conference on*. IEEE, 2016.
358. Ben Kybartas and Rafael Bidarra. A semantic foundation for mixed-initiative computational storytelling. In *Interactive Storytelling*, pages 162–169. Springer, 2015.
359. Alexandros Labrinidis and Hosagrahar V. Jagadish. Challenges and opportunities with big data. *Proceedings of the VLDB Endowment*, 5(12):2032–2033, 2012.
360. John Laird and Michael van Lent. Human-level AI's killer application: Interactive computer games. *AI Magazine*, 22(2):15, 2001.
361. G. B. Langley and H. Shepperd. The visual analogue scale: its use in pain measurement. *Rheumatology International*, 5(4):145–148, 1985.
362. Frank Lantz, Aaron Isaksen, Alexander Jaffe, Andy Nealen, and Julian Togelius. Depth in strategic games. In *Proceedings of the AAAI WNAIG Workshop*, 2017.
363. Pier Luca Lanzi, Wolfgang Stolzmann, and Stewart W. Wilson. *Learning classifier systems: from foundations to applications*. Springer, 2003.
364. Richard S. Lazarus. *Emotion and adaptation*. Oxford University Press, 1991.
365. Nicole Lazzaro. Why we play games: Four keys to more emotion without story. Technical report, XEO Design Inc., 2004.
366. Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.
367. David Lee and Mihalis Yannakakis. Principles and methods of testing finite state machines—a survey. *Proceedings of the IEEE*, 84(8):1090–1123, 1996.
368. Alan Levinovitz. The mystery of Go, the ancient game that computers still can't win. *Wired Magazine*, 2014.
369. Mike Lewis and Kevin Dill. Game AI appreciation, revisited. In *Game AI Pro 2: Collected Wisdom of Game AI Professionals*, pages 3–18. AK Peters/CRC Press, 2015.
370. Boyang Li, Stephen Lee-Urban, Darren Scott Appling, and Mark O. Riedl. Crowdsourcing narrative intelligence. *Advances in Cognitive Systems*, 2(1), 2012.
371. Antonios Liapis. Creativity facet orchestration: the whys and the hows. *Artificial and Computational Intelligence in Games: Integration; Dagstuhl Follow-Ups*, 2015.
372. Antonios Liapis. Mixed-initiative Creative Drawing with webIconoscope. In *Proceedings of the 6th International Conference on Computational Intelligence in Music, Sound, Art and Design. (EvoMusArt)*. Springer, 2017.
373. Antonios Liapis, Héctor P. Martinez, Julian Togelius, and Georgios N. Yannakakis. Transforming exploratory creativity with DeLeNoX. In *Proceedings of the Fourth International Conference on Computational Creativity*, pages 56–63, 2013.
374. Antonios Liapis, Gillian Smith, and Noor Shaker. Mixed-initiative content creation. In *Procedural Content Generation in Games*, pages 195–214. Springer, 2016.
375. Antonios Liapis and Georgios N. Yannakakis. Boosting computational creativity with human interaction in mixed-initiative co-creation tasks. In *Proceedings of the ICCC Workshop on Computational Creativity and Games*, 2016.

376. Antonios Liapis, Georgios N. Yannakakis, and Julian Togelius. Neuroevolutionary constrained optimization for content creation. In *Computational Intelligence and Games (CIG), 2011 IEEE Conference on*, pages 71–78. IEEE, 2011.
377. Antonios Liapis, Georgios N. Yannakakis, and Julian Togelius. Adapting models of visual aesthetics for personalized content creation. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(3):213–228, 2012.
378. Antonios Liapis, Georgios N. Yannakakis, and Julian Togelius. Designer modeling for personalized game content creation tools. In *Proceedings of the AIIDE Workshop on Artificial Intelligence & Game Aesthetics*, 2013.
379. Antonios Liapis, Georgios N. Yannakakis, and Julian Togelius. Sentient Sketchbook: Computer-aided game level authoring. In *Proceedings of ACM Conference on Foundations of Digital Games*, pages 213–220, 2013.
380. Antonios Liapis, Georgios N. Yannakakis, and Julian Togelius. Sentient World: Human-Based Procedural Cartography. In *Evolutionary and Biologically Inspired Music, Sound, Art and Design*, pages 180–191. Springer, 2013.
381. Antonios Liapis, Georgios N. Yannakakis, and Julian Togelius. Computational Game Creativity. In *Proceedings of the Fifth International Conference on Computational Creativity*, pages 285–292, 2014.
382. Antonios Liapis, Georgios N. Yannakakis, and Julian Togelius. Constrained novelty search: A study on game content generation. *Evolutionary Computation*, 23(1):101–129, 2015.
383. Vladimir Lifschitz. Answer set programming and plan generation. *Artificial Intelligence*, 138(1-2):39–54, 2002.
384. Rensis Likert. A technique for the measurement of attitudes. *Archives of Psychology*, 140:1–55, 1932.
385. Chong-U Lim, Robin Baumgarten, and Simon Colton. Evolving behaviour trees for the commercial game DEFCON. In *European Conference on the Applications of Evolutionary Computation*, pages 100–110. Springer, 2010.
386. Chong-U Lim and D. Fox Harrell. Revealing social identity phenomena in videogames with archetypal analysis. In *Proceedings of the 6th International AISB Symposium on AI and Games*, 2015.
387. Aristid Lindenmayer. Mathematical models for cellular interactions in development I. Filaments with one-sided inputs. *Journal of Theoretical Biology*, 18(3):280–299, 1968.
388. R. L. Linn and N. E. Gronlund. *Measurement and assessment in teaching*. Prentice-Hall, 2000.
389. Nir Lipovetzky and Hector Geffner. Width-based algorithms for classical planning: New results. In *Proceedings of the Twenty-first European Conference on Artificial Intelligence*, pages 1059–1060. IOS Press, 2014.
390. Nir Lipovetzky, Miquel Ramirez, and Hector Geffner. Classical Planning with Simulators: Results on the Atari Video Games. In *Proceedings of IJCAI*, pages 1610–1616, 2015.
391. Maria Teresa Llano, Michael Cook, Christian Guckelsberger, Simon Colton, and Rose Hepworth. Towards the automatic generation of fictional ideas for games. In *Experimental AI in Games (EXAG14), a Workshop collocated with the Tenth Annual AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE14)*. AAAI Publications, 2014.
392. Daniele Loiacono, Pier Luca Lanzi, Julian Togelius, Enrique Onieva, David A. Pelta, Martin V. Butz, Thies D. Lönneker, Luigi Cardamone, Diego Perez, Yago Sáez, Mike Preuss, and Jan Quadflieg. The 2009 simulated car racing championship. *Computational Intelligence and AI in Games, IEEE Transactions on*, 2(2):131–147, 2010.
393. Daniele Loiacono, Julian Togelius, Pier Luca Lanzi, Leonard Kinnaird-Heether, Simon M. Lucas, Matt Simmerson, Diego Perez, Robert G. Reynolds, and Yago Saez. The WCCI 2008 simulated car racing competition. In *IEEE Symposium on Computational Intelligence and Games*, pages 119–126. IEEE, 2008.
394. Phil Lopes, Antonios Liapis, and Georgios N. Yannakakis. Sonancia: Sonification of procedurally generated game levels. In *Proceedings of the ICCC workshop on Computational Creativity & Games*, 2015.

395. Phil Lopes, Antonios Liapis, and Georgios N. Yannakakis. Framing tension for game generation. In *Proceedings of the Seventh International Conference on Computational Creativity*, 2016.
396. Phil Lopes, Antonios Liapis, and Georgios N. Yannakakis. Modelling affect for horror soundscapes. *IEEE Transactions on Affective Computing*, 2017.
397. Phil Lopes, Georgios N. Yannakakis, and Antonios Liapis. RankTrace: Relative and Unbounded Affect Annotation. In *Affective Computing and Intelligent Interaction (ACII), 2017 International Conference on*, 2017.
398. Ricardo Lopes and Rafael Bidarra. Adaptivity challenges in games and simulations: a survey. *Computational Intelligence and AI in Games, IEEE Transactions on*, 3(2):85–99, 2011.
399. Sandy Louchart, Ruth Aylett, Joao Dias, and Ana Paiva. Unscripted narrative for affectively driven characters. In *AIIDE*, pages 81–86, 2005.
400. Nathaniel Love, Timothy Hinrichs, David Haley, Eric Schkufza, and Michael Genesereth. General game playing: Game description language specification. Technical Report LG-2006-01, Stanford Logic Group, Computer Science Department, Stanford University, 2008.
401. A. Bryan Loyall and Joseph Bates. Personality-rich believable agents that use language. In *Proceedings of the First International Conference on Autonomous Agents*, pages 106–113. ACM, 1997.
402. Feiyu Lu, Kaito Yamamoto, Luis H. Nomura, Syunsuke Mizuno, YoungMin Lee, and Ruck Thawonmas. Fighting game artificial intelligence competition platform. In *Consumer Electronics (GCCE), 2013 IEEE 2nd Global Conference on*, pages 320–323. IEEE, 2013.
403. Simon M. Lucas. Evolving a Neural Network Location Evaluator to Play Ms. Pac-Man. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games*, pages 203–210, 2005.
404. Simon M. Lucas. Ms Pac-Man competition. *ACM SIGEVolution*, 2(4):37–38, 2007.
405. Simon M. Lucas. Computational intelligence and games: Challenges and opportunities. *International Journal of Automation and Computing*, 5(1):45–57, 2008.
406. Simon M. Lucas and Graham Kendall. Evolutionary computation and games. *Computational Intelligence Magazine, IEEE*, 1(1):10–18, 2006.
407. Simon M. Lucas, Michael Mateas, Mike Preuss, Pieter Spronck, and Julian Togelius. Artificial and Computational Intelligence in Games (Dagstuhl Seminar 12191). *Dagstuhl Reports*, 2(5):43–70, 2012.
408. Simon M. Lucas and T. Jeff Reynolds. Learning finite-state transducers: Evolution versus heuristic state merging. *IEEE Transactions on Evolutionary Computation*, 11(3):308–325, 2007.
409. Jeremy Ludwig and Art Farley. A learning infrastructure for improving agent performance and game balance. In Georgios N. Yannakakis and John Hallam, editors, *Proceedings of the AIIDE'07 Workshop on Optimizing Player Satisfaction, Technical Report WS-07-01*, pages 7–12. AAAI Press, 2007.
410. Kevin Lynch. *The Image of the City*. MIT Press, 1960.
411. James MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, number 14, pages 281–297. Oakland, CA, USA, 1967.
412. Brian Magerko. Story representation and interactive drama. In *AIIDE*, pages 87–92, 2005.
413. Brendan Maher. Can a video game company tame toxic behaviour? *Nature*, 531(7596):568–571, 2016.
414. Tobias Mahlmann, Anders Drachen, Julian Togelius, Alessandro Canossa, and Georgios N. Yannakakis. Predicting player behavior in Tomb Raider: Underworld. In *Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games*, pages 178–185. IEEE, 2010.
415. Tobias Mahlmann, Julian Togelius, and Georgios N. Yannakakis. Modelling and evaluation of complex scenarios with the strategy game description language. In *Computational Intelligence and Games (CIG), 2011 IEEE Conference on*, pages 174–181. IEEE, 2011.

416. Tobias Mahlmann, Julian Togelius, and Georgios N. Yannakakis. Evolving card sets towards balancing Dominion. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2012.
417. Kevin Majchrzak, Jan Quadflieg, and Günter Rudolph. Advanced dynamic scripting for fighting game AI. In *International Conference on Entertainment Computing*, pages 86–99. Springer, 2015.
418. Nikos Malandrakis, Alexandros Potamianos, Georgios Evangelopoulos, and Athanasia Zlatintsi. A supervised approach to movie emotion tracking. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 2376–2379. IEEE, 2011.
419. Thomas W. Malone. What makes computer games fun? *Byte*, 6:258–277, 1981.
420. Regan L. Mandryk and M. Stella Atkins. A fuzzy physiological approach for continuously modeling emotion during interaction with play technologies. *International Journal of Human-Computer Studies*, 65(4):329–347, 2007.
421. Regan L. Mandryk, Kori M. Inkpen, and Thomas W. Calvert. Using psychophysiological techniques to measure user experience with entertainment technologies. *Behaviour & Information Technology*, 25(2):141–158, 2006.
422. Jacek Mandziuk. Computational intelligence in mind games. *Challenges for Computational Intelligence*, 63:407–442, 2007.
423. Jacek Mandziuk. *Knowledge-free and learning-based methods in intelligent game playing*. Springer, 2010.
424. Benjamin Mark, Tudor Berechet, Tobias Mahlmann, and Julian Togelius. Procedural Generation of 3D Caves for Games on the GPU. In *Proceedings of the Conference on the Foundations of Digital Games (FDG)*, 2015.
425. Dave Mark. *Behavioral Mathematics for game AI*. Charles River Media, 2009.
426. Dave Mark and Kevin Dill. Improving AI decision modeling through utility theory. In *Game Developers Conference*, 2010.
427. Gloria Mark, Yiran Wang, and Melissa Niiya. Stress and multitasking in everyday college life: An empirical study of online activity. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 41–50, 2014.
428. Stacy Marsella, Jonathan Gratch, and Paolo Petta. Computational models of emotion. A *Blueprint for Affective Computing—A sourcebook and manual*, 11(1):21–46, 2010.
429. Chris Martens. Cepre: A language for modeling generative interactive systems. In *Eleventh Artificial Intelligence and Interactive Digital Entertainment Conference*, 2015.
430. Héctor P. Martínez, Yoshua Bengio, and Georgios N. Yannakakis. Learning deep physiological models of affect. *Computational Intelligence Magazine, IEEE*, 9(1):20–33, 2013.
431. Héctor P. Martínez, Maurizio Garbarino, and Georgios N. Yannakakis. Generic physiological features as predictors of player experience. In *Affective Computing and Intelligent Interaction*, pages 267–276. Springer, 2011.
432. Héctor P. Martínez, Kenneth Hullett, and Georgios N. Yannakakis. Extending neuro-evolutionary preference learning through player modeling. In *Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games*, pages 313–320. IEEE, 2010.
433. Héctor P. Martínez and Georgios N. Yannakakis. Genetic search feature selection for affective modeling: a case study on reported preferences. In *Proceedings of the 3rd International Workshop on Affective Interaction in Natural Environments*, pages 15–20. ACM, 2010.
434. Héctor P. Martínez and Georgios N. Yannakakis. Mining multimodal sequential patterns: a case study on affect detection. In *Proceedings of International Conference on Multimodal Interfaces (ICMI)*, pages 3–10. ACM, 2011.
435. Héctor P. Martínez and Georgios N. Yannakakis. Deep multimodal fusion: Combining discrete events and continuous signals. In *Proceedings of the 16th International Conference on Multimodal Interaction*, pages 34–41. ACM, 2014.
436. Héctor P. Martínez, Georgios N. Yannakakis, and John Hallam. Don't Classify Ratings of Affect; Rank them! *IEEE Transactions on Affective Computing*, 5(3):314–326, 2014.

437. Giovanna Martinez-Arellano, Richard Cant, and David Woods. Creating AI Characters for Fighting Games using Genetic Programming. *IEEE Transactions on Computational Intelligence and AI in Games*, 2016.
438. Michael Mateas. *Interactive Drama, Art and Artificial Intelligence*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA, 2002.
439. Michael Mateas. Expressive AI: Games and Artificial Intelligence. In *DIGRA Conference*, 2003.
440. Michael Mateas and Andrew Stern. A behavior language for story-based believable agents. *IEEE Intelligent Systems*, 17(4):39–47, 2002.
441. Michael Mateas and Andrew Stern. Façade: An experiment in building a fully-realized interactive drama. In *Game Developers Conference*, 2003.
442. Michael Mauderer, Simone Conte, Miguel A. Nacenta, and Dhanraj Vishwanath. Depth perception with gaze-contingent depth of field. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 217–226, 2014.
443. John D. Mayer and Peter Salovey. The intelligence of emotional intelligence. *Intelligence*, 17(4):433–442, 1993.
444. Allan Mazur, Elizabeth J. Susman, and Sandy Edelbrock. Sex difference in testosterone response to a video game contest. *Evolution and Human Behavior*, 18(5):317–326, 1997.
445. Andrew McAfee, Erik Brynjolfsson, Thomas H. Davenport, D. J. Patil, and Dominic Barton. Big data. *The management revolution. Harvard Bus Rev*, 90(10):61–67, 2012.
446. John McCarthy. Partial formalizations and the Lemmings game. Technical report, Stanford University, 1998.
447. Josh McCoy, Mike Treanor, Ben Samuel, Michael Mateas, and Noah Wardrip-Fruin. Prom week: social physics as gameplay. In *Proceedings of the 6th International Conference on Foundations of Digital Games*, pages 319–321. ACM, 2011.
448. Josh McCoy, Mike Treanor, Ben Samuel, Aaron A. Reed, Noah Wardrip-Fruin, and Michael Mateas. Prom week. In *Proceedings of the International Conference on the Foundations of Digital Games*, pages 235–237. ACM, 2012.
449. Robert R. McCrae and Paul T. Costa Jr. A five-factor theory of personality. *Handbook of personality: Theory and research*, 2:139–153, 1999.
450. Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5(4):115–133, 1943.
451. Scott W. McQuiggan, Sunyoung Lee, and James C. Lester. Early prediction of student frustration. In *Proceedings of International Conference on Affective Computing and Intelligent Interaction*, pages 698–709. Springer, 2007.
452. Scott W. McQuiggan, Bradford W. Mott, and James C. Lester. Modeling self-efficacy in intelligent tutoring systems: An inductive approach. *User Modeling and User-Adapted Interaction*, 18(1):81–123, 2008.
453. Andre Mendes, Julian Togelius, and Andy Nealen. Hyper-heuristic general video game playing. In *Computational Intelligence and Games (CIG), 2016 IEEE Conference on*. IEEE, 2016.
454. Daniel S. Messinger, Tricia D. Cassel, Susan I. Acosta, Zara Ambadar, and Jeffrey F. Cohn. Infant smiling dynamics and perceived positive emotion. *Journal of Nonverbal Behavior*, 32(3):133–155, 2008.
455. Angeliki Metallinou and Shrikanth Narayanan. Annotation and processing of continuous emotional attributes: Challenges and opportunities. In *10th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*. IEEE, 2013.
456. Zbigniew Michalewicz. Do not kill unfeasible individuals. In *Proceedings of the Fourth Intelligent Information Systems Workshop*, pages 110–123, 1995.
457. Risto Miikkulainen, Bobby D. Bryant, Ryan Cornelius, Igor V. Karpov, Kenneth O. Stanley, and Chern Han Yong. Computational intelligence in games. *Computational Intelligence: Principles and Practice*, pages 155–191, 2006.
458. Benedikte Mikkelsen, Christoffer Holmgård, and Julian Togelius. Ethical Considerations for Player Modeling. In *Proceedings of the AAAI WNAIG workshop*, 2017.

459. Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.
460. George A. Miller. The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological Review*, 63(2):81, 1956.
461. Ian Millington and John Funge. *Artificial intelligence for games*. CRC Press, 2009.
462. Talya Miron-Shatz, Arthur Stone, and Daniel Kahneman. Memories of yesterday’s emotions: Does the valence of experience affect the memory-experience gap? *Emotion*, 9(6):885, 2009.
463. Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1928–1937, 2016.
464. Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.
465. Mathew Monfort, Matthew Johnson, Aude Oliva, and Katja Hofmann. Asynchronous data aggregation for training end to end visual control networks. In *Proceedings of the 16th Conference on Autonomous Agents and Multi-Agent Systems*, pages 530–537. International Foundation for Autonomous Agents and Multiagent Systems, May 2017.
466. Nick Montfort and Ian Bogost. *Racing the beam: The Atari video computer system*. MIT Press, 2009.
467. Matej Moravčík, Martin Schmid, Neil Burch, Vilim Lisý, Dustin Morrill, Nolan Bard, Trevor Davis, Kevin Waugh, Michael Johanson, and Michael Bowling. Deepstack: Expert-level artificial intelligence in no-limit poker. *arXiv preprint arXiv:1701.01724*, 2017.
468. Jon D. Morris. Observations: SAM: The self-assessment manikin—An efficient cross-cultural measurement of emotional response. *Journal of Advertising Research*, 35(6):63–68, 1995.
469. Jorge Munoz, Georgios N. Yannakakis, Fiona Mulvey, Dan Witzner Hansen, German Gutierrez, and Araceli Sanchis. Towards gaze-controlled platform games. In *Computational Intelligence and Games (CIG), 2011 IEEE Conference on*, pages 47–54. IEEE, 2011.
470. Hector Munoz-Avila, Christian Bauckhage, Michal Bida, Clare Bates Congdon, and Graham Kendall. Learning and Game AI. *Dagstuhl Follow-Ups*, 6, 2013.
471. Roger B. Myerson. Game theory: analysis of conflict. 1991. *Cambridge: Mass, Harvard University*.
472. Roger B. Myerson. *Game theory*. Harvard University Press, 2013.
473. Lennart Nacke and Craig A. Lindley. Flow and immersion in first-person shooters: measuring the player’s gameplay experience. In *Proceedings of the 2008 Conference on Future Play: Research, Play, Share*, pages 81–88. ACM, 2008.
474. Frederik Nagel, Reinhard Kopiez, Oliver Grewe, and Eckart Altenmüller. Emujoy: Software for continuous measurement of perceived emotions in music. *Behavior Research Methods*, 39(2):283–290, 2007.
475. Karthik Narasimhan, Tejas Kulkarni, and Regina Barzilay. Language understanding for text-based games using deep reinforcement learning. *arXiv preprint arXiv:1506.08941*, 2015.
476. Alexander Nareyek. Intelligent agents for computer games. In T.A. Marsland and I. Frank, editors, *Computers and Games, Second International Conference, CG 2002*, pages 414–422, 2002.
477. Alexander Nareyek. Game AI is dead. Long live game AI! *IEEE Intelligent Systems*, (1):9–11, 2007.
478. John F. Nash. Equilibrium points in n-person games. In *Proceedings of the National Academy of Sciences*, number 1, pages 48–49, 1950.
479. Steve Nebel, Sascha Schneider, and Günter Daniel Rey. Mining learning and crafting scientific experiments: a literature review on the use of Minecraft in education and research. *Journal of Educational Technology & Society*, 19(2):355, 2016.

480. Graham Nelson. Natural language, semantic analysis, and interactive fiction. *IF Theory Reader*, 141, 2006.
481. Mark J. Nelson. Game Metrics Without Players: Strategies for Understanding Game Artifacts. In *AIIDE Workshop on Artificial Intelligence in the Game Design Process*, 2011.
482. Mark J. Nelson, Simon Colton, Edward J. Powley, Swen E. Gaudl, Peter Ivey, Rob Saunders, Blanca Pérez Ferrer, and Michael Cook. Mixed-initiative approaches to on-device mobile game design. In *Proceedings of the CHI Workshop on Mixed-Initiative Creative Interfaces*, 2017.
483. Mark J. Nelson and Michael Mateas. Search-Based Drama Management in the Interactive Fiction Anchorhead. In *Proceedings of the First Artificial Intelligence and Interactive Digital Entertainment Conference*, pages 99–104, 2005.
484. Mark J. Nelson and Michael Mateas. An interactive game-design assistant. In *Proceedings of the 13th International Conference on Intelligent User Interfaces*, pages 90–98, 2008.
485. Mark J. Nelson and Adam M. Smith. ASP with applications to mazes and levels. In *Procedural Content Generation in Games*, pages 143–157. Springer, 2016.
486. Mark J. Nelson, Julian Togelius, Cameron Browne, and Michael Cook. Rules and mechanics. In *Procedural Content Generation in Games*, pages 99–121. Springer, 2016.
487. John Von Neumann. *Theory of Self-Reproducing Automata*. University of Illinois Press, Champaign, IL, USA, 1966.
488. Truong-Huy D. Nguyen, Shree Subramanian, Magy Seif El-Nasr, and Alessandro Canossa. Strategy Detection in Wuzzit: A Decision Theoretic Approach. In *International Conference on Learning Science—Workshop on Learning Analytics for Learning and Becoming a Practice*, 2014.
489. Jakob Nielsen. Usability 101: Introduction to usability, 2003. Available at <http://www.useit.com/alertbox/20030825.html>.
490. Jon Lau Nielsen, Benjamin Fedder Jensen, Tobias Mahlmann, Julian Togelius, and Georgios N. Yannakakis. AI for General Strategy Game Playing. *Handbook of Digital Games*, pages 274–304, 2014.
491. Thorbjørn S. Nielsen, Gabriella A. B. Barros, Julian Togelius, and Mark J. Nelson. General Video Game Evaluation Using Relative Algorithm Performance Profiles. In *Applications of Evolutionary Computation*, pages 369–380. Springer, 2015.
492. Thorbjørn S. Nielsen, Gabriella A. B. Barros, Julian Togelius, and Mark J. Nelson. Towards generating arcade game rules with VGDL. In *Proceedings of the 2015 IEEE Conference on Computational Intelligence and Games*, 2015.
493. Anton Nijholt. BCI for games: A state of the art survey. In *Entertainment Computing-ICEC 2008*, pages 225–228. Springer, 2009.
494. Nils J. Nilsson. Shakey the robot. Technical report, DTIC Document, 1984.
495. Kai Ninomiya, Mubbasis Kapadia, Alexander Shoulson, Francisco Garcia, and Norman Badler. Planning approaches to constraint-aware navigation in dynamic environments. *Computer Animation and Virtual Worlds*, 26(2):119–139, 2015.
496. Stefano Nolfi and Dario Floreano. *Evolutionary robotics: The biology, intelligence, and technology of self-organizing machines*. MIT Press, 2000.
497. David G. Novick and Stephen Sutton. What is mixed-initiative interaction. In *Proceedings of the AAAI Spring Symposium on Computational Models for Mixed Initiative Interaction*, pages 114–116, 1997.
498. Gabriela Ochoa. On genetic algorithms and Lindenmayer systems. In *Parallel Problem Solving from Nature—PPSN V*, pages 335–344. Springer, 1998.
499. Jacob Kaae Olesen, Georgios N. Yannakakis, and John Hallam. Real-time challenge balance in an RTS game using rtNEAT. In *Computational Intelligence and Games, 2008. CIG'08. IEEE Symposium On*, pages 87–94. IEEE, 2008.
500. Jacob Olsen. Realtime procedural terrain generation. 2004.
501. Peter Thorup Ølsted, Benjamin Ma, and Sebastian Risi. Interactive evolution of levels for a competitive multiplayer FPS. In *Evolutionary Computation (CEC), 2015 IEEE Congress on*, pages 1527–1534. IEEE, 2015.

502. Cathy O’Neil. *Weapons of math destruction: How big data increases inequality and threatens democracy*. Crown Publishing Group (NY), 2016.
503. Santiago Ontañón. The combinatorial multi-armed bandit problem and its application to real-time strategy games. In *Ninth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2013.
504. Santiago Ontañón, Gabriel Synnaeve, Alberto Uriarte, Florian Richoux, David Churchill, and Mike Preuss. A survey of real-time strategy game AI research and competition in StarCraft. *IEEE Transactions on Computational Intelligence and AI in Games*, 5(4):293–311, 2013.
505. Santiago Ontañón, Gabriel Synnaeve, Alberto Uriarte, Florian Richoux, David Churchill, and Mike Preuss. RTS AI: Problems and Techniques. In *Encyclopedia of Computer Graphics and Games*. Springer, 2015.
506. Jeff Orkin. Applying goal-oriented action planning to games. *AI game programming wisdom*, 2:217–228, 2003.
507. Jeff Orkin. Three states and a plan: the AI of F.E.A.R. In *Game Developers Conference*, 2006.
508. Jeff Orkin and Deb Roy. The restaurant game: Learning social behavior and language from thousands of players online. *Journal of Game Development*, 3(1):39–60, 2007.
509. Mauricio Orozco, Juan Silva, Abdulmotaleb El Saddik, and Emil Petriu. The role of haptics in games. In *Haptics Rendering and Applications*. InTech, 2012.
510. Brian O’Neill and Mark Riedl. Emotion-driven narrative generation. In *Emotion in Games: Theory and Praxis*, pages 167–180. Springer, 2016.
511. Juan Ortega, Noor Shaker, Julian Togelius, and Georgios N. Yannakakis. Imitating human playing styles in Super Mario Bros. *Entertainment Computing*, 4(2):93–104, 2013.
512. Andrew Ortony, Gerald L. Clore, and Allan Collins. *The cognitive structure of emotions*. Cambridge University Press, 1990.
513. Martin J. Osborne. *An introduction to game theory*. Oxford University Press, 2004.
514. Alexander Osherenko. *Opinion Mining and Lexical Affect Sensing. Computer-aided analysis of opinions and emotions in texts*. PhD thesis, University of Augsburg, 2010.
515. Seth Ovadia. Ratings and rankings: Reconsidering the structure of values and their measurement. *International Journal of Social Research Methodology*, 7(5):403–414, 2004.
516. Ana Paiva, Joao Dias, Daniel Sobral, Ruth Aylett, Polly Sobrepeirez, Sarah Woods, Carsten Zoll, and Lynne Hall. Caring for agents and agents that care: Building empathic relations with synthetic agents. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 194–201. IEEE Computer Society, 2004.
517. Bo Pang and Lillian Lee. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1–2):1–135, 2008.
518. Matt Parker and Bobby D. Bryant. Visual control in Quake II with a cyclic controller. In *Computational Intelligence and Games, 2008. CIG’08. IEEE Symposium On*, pages 151–158. IEEE, 2008.
519. Matt Parker and Bobby D. Bryant. Neurovisual control in the Quake II environment. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):44–54, 2012.
520. Chris Pedersen, Julian Togelius, and Georgios N. Yannakakis. Modeling Player Experience in Super Mario Bros. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games*, pages 132–139. IEEE, 2009.
521. Chris Pedersen, Julian Togelius, and Georgios N. Yannakakis. Modeling Player Experience for Content Creation. *IEEE Transactions on Computational Intelligence and AI in Games*, 2(1):54–67, 2010.
522. Barney Pell. *Strategy generation and evaluation for meta-game playing*. PhD thesis, University of Cambridge, 1993.
523. Peng Peng, Quan Yuan, Ying Wen, Yaodong Yang, Zhenkun Tang, Haitao Long, and Jun Wang. Multiagent Bidirectionally-Coordinated Nets for Learning to Play StarCraft Combat Games. *arXiv preprint arXiv:1703.10069*, 2017.
524. Tom Pepels, Mark H. M. Winands, and Marc Lanctot. Real-time Monte Carlo tree search in Ms Pac-Man. *IEEE Transactions on Computational Intelligence and AI in Games*, 6(3):245–257, 2014.

525. Diego Perez, Edward J. Powley, Daniel Whitehouse, Philipp Rohlfshagen, Spyridon Samothrakis, Peter I. Cowling, and Simon M. Lucas. Solving the physical traveling salesman problem: Tree search and macro actions. *IEEE Transactions on Computational Intelligence and AI in Games*, 6(1):31–45, 2014.
526. Diego Perez, Spyridon Samothrakis, Simon Lucas, and Philipp Rohlfshagen. Rolling horizon evolution versus tree search for navigation in single-player real-time games. In *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*, pages 351–358. ACM, 2013.
527. Diego Perez-Liebana, Spyridon Samothrakis, Julian Togelius, Tom Schaul, and Simon M. Lucas. General video game AI: Competition, challenges and opportunities. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, 2016.
528. Diego Perez-Liebana, Spyridon Samothrakis, Julian Togelius, Tom Schaul, Simon M. Lucas, Adrien Couëtoux, Jerry Lee, Chong-U Lim, and Tommy Thompson. The 2014 general video game playing competition. *IEEE Transactions on Computational Intelligence and AI in Games*, 8(3):229–243, 2016.
529. Ken Perlin. An image synthesizer. *ACM SIGGRAPH Computer Graphics*, 19(3):287–296, 1985.
530. Rosalind W. Picard. *Affective Computing*. MIT Press, Cambridge, MA, 1997.
531. Grant Pickett, Foad Khosmood, and Allan Fowler. Automated generation of conversational non player characters. In *Eleventh Artificial Intelligence and Interactive Digital Entertainment Conference*, 2015.
532. Michele Pirovano. The use of Fuzzy Logic for Artificial Intelligence in Games. Technical report, University of Milano, Milano, 2012.
533. Jacques Pitrat. Realization of a general game-playing program. In *IFIP congress (2)*, pages 1570–1574, 1968.
534. Isabella Poggi, Catherine Pelachaud, Fiorella de Rosis, Valeria Carofiglio, and Berardina De Carolis. GRETA. A believable embodied conversational agent. In *Multimodal intelligent information presentation*, pages 3–25. Springer, 2005.
535. Mihai Polceanu. Mirrorbot: Using human-inspired mirroring behavior to pass a Turing test. In *Computational Intelligence in Games (CIG), 2013 IEEE Conference on*. IEEE, 2013.
536. Riccardo Poli, William B. Langdon, and Nicholas F. McPhee. *A field guide to genetic programming*. 2008. Published via <http://lulu.com> and freely available at <http://www.gp-field-guide.org.uk> (With contributions by J. R. Koza).
537. Jordan B. Pollack and Alan D. Blair. Co-evolution in the successful learning of backgammon strategy. *Machine learning*, 32(3):225–240, 1998.
538. Jordan B. Pollack, Alan D. Blair, and Mark Land. Coevolution of a backgammon player. In *Artificial Life V: Proceedings of the Fifth International Workshop on the Synthesis and Simulation of Living Systems*, pages 92–98. Cambridge, MA: The MIT Press, 1997.
539. Jonathan Posner, James A. Russell, and Bradley S. Peterson. The circumplex model of affect: An integrative approach to affective neuroscience, cognitive development, and psychopathology. *Development and psychopathology*, 17(03):715–734, 2005.
540. David Premack and Guy Woodruff. Does the chimpanzee have a theory of mind? *Behavioral and brain sciences*, 1(04):515–526, 1978.
541. Mike Preuss, Daniel Kozakowski, Johan Hagelbäck, and Heike Trautmann. Reactive strategy choice in StarCraft by means of Fuzzy Control. In *Computational Intelligence in Games (CIG), 2013 IEEE Conference on*. IEEE, 2013.
542. Przemyslaw Prusinkiewicz and Aristid Lindenmayer. *The algorithmic beauty of plants*. Springer, 1990.
543. Jan Quadflieg, Mike Preuss, and Günter Rudolph. Driving as a human: a track learning based adaptable architecture for a car racing controller. *Genetic Programming and Evolvable Machines*, 15(4):433–476, 2014.
544. J. Ross Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.
545. J. Ross Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.
546. Steve Rabin. *AI Game Programming Wisdom*. Charles River Media, Inc., 2002.

547. Steve Rabin. *AI Game Programming Wisdom 2*. Charles River Media, Inc., 2003.
548. Steve Rabin. *AI Game Programming Wisdom 3*. Charles River Media, Inc., 2006.
549. Steve Rabin. *AI Game Programming Wisdom 4*. Nelson Education, 2014.
550. Steve Rabin and Nathan Sturtevant. Pathfinding Architecture Optimizations. In *Game AI Pro: Collected Wisdom of Game AI Professionals*. CRC Press, 2013.
551. Steve Rabin and Nathan Sturtevant. Combining Bounding Boxes and JPS to Prune Grid Pathfinding. In *AAAI Conference on Artificial Intelligence*, 2016.
552. Steven Rabin. *Game AI Pro: Collected Wisdom of Game AI Professionals*. CRC Press, 2013.
553. Steven Rabin. *Game AI Pro 2: Collected Wisdom of Game AI Professionals*. CRC Press, 2015.
554. William L. Raffe, Fabio Zambetta, and Xiaodong Li. A survey of procedural terrain generation techniques using evolutionary algorithms. In *IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2012.
555. Judith Ramey, Ted Boren, Elisabeth Cuddihy, Joe Dumas, Zhiwei Guan, Maaike J. van den Haak, and Menno D. T. De Jong. Does think aloud work? How do we know? In *CHI'06 Extended Abstracts on Human Factors in Computing Systems*, pages 45–48. ACM, 2006.
556. Pramila Rani, Nilanjan Sarkar, and Changchun Liu. Maintaining optimal challenge in computer games through real-time physiological feedback. In *Proceedings of the 11th International Conference on Human Computer Interaction*, pages 184–192, 2005.
557. Jakob Rasmussen. *Are Behavior Trees a Thing of the Past?* Gamasutra, 2016.
558. Niklas Ravaja, Timo Saari, Mikko Salminen, Jari Laarni, and Kari Kallinen. Phasic emotional reactions to video game events: A psychophysiological investigation. *Media Psychology*, 8(4):343–367, 2006.
559. Genaro Rebolledo-Mendez, Ian Dunwell, Erika Martínez-Mirón, María Dolores Vargas-Cerdán, Sara De Freitas, Fotis Liarokapis, and Alma R. García-Gaona. Assessing Neurosky’s usability to detect attention levels in an assessment exercise. *Human-Computer Interaction. New Trends*, pages 149–158, 2009.
560. Jochen Renz, Xiaoyu Ge, Stephen Gould, and Peng Zhang. The Angry Birds AI Competition. *AI Magazine*, 36(2):85–87, 2015.
561. Antonio Ricciardi and Patrick Thill. Adaptive AI for Fighting Games. Technical report, Stanford University, 2008.
562. Mark O. Riedl and Vadim Bulitko. Interactive narrative: An intelligent systems approach. *AI Magazine*, 34(1):67, 2012.
563. Mark O. Riedl and Andrew Stern. Believable agents and intelligent story adaptation for interactive storytelling. *Technologies for Interactive Digital Storytelling and Entertainment*, pages 1–12, 2006.
564. Mark O. Riedl and Alexander Zook. AI for game production. In *IEEE Conference on Computational Intelligence in Games (CIG)*. IEEE, 2013.
565. Sebastian Risi, Joel Lehman, David B. D'Ambrosio, Ryan Hall, and Kenneth O. Stanley. Combining Search-Based Procedural Content Generation and Social Gaming in the Petalz Video Game. In *Proceedings of AIIDE*, 2012.
566. Sebastian Risi, Joel Lehman, David B. D'Ambrosio, Ryan Hall, and Kenneth O. Stanley. Petalz: Search-based procedural content generation for the casual gamer. *IEEE Transactions on Computational Intelligence and AI in Games*, 8(3):244–255, 2016.
567. Sebastian Risi and Julian Togelius. Neuroevolution in games: State of the art and open challenges. *IEEE Transactions on Computational Intelligence and AI in Games*, 9(1):25–41, 2017.
568. David L. Roberts, Harikrishna Narayanan, and Charles L. Isbell. Learning to influence emotional responses for interactive storytelling. In *Proceedings of the 2009 AAAI Symposium on Intelligent Narrative Technologies II*, 2009.
569. Glen Robertson and Ian D. Watson. A review of real-time strategy game AI. *AI Magazine*, 35(4):75–104, 2014.
570. Glen Robertson and Ian D. Watson. An Improved Dataset and Extraction Process for StarCraft AI. In *FLAIRS Conference*, 2014.

571. Michael D. Robinson and Gerald L. Clore. Belief and feeling: evidence for an accessibility model of emotional self-report. *Psychological Bulletin*, 128(6):934, 2002.
572. Jennifer Robison, Scott McQuiggan, and James Lester. Evaluating the consequences of affective feedback in intelligent tutoring systems. In *Proceedings of International Conference on Affective Computing and Intelligent Interaction (ACII)*. IEEE, 2009.
573. Philipp Rohlfschagen, Jialin Liu, Diego Perez-Liebana, and Simon M. Lucas. Pac-Man Conquers Academia: Two Decades of Research Using a Classic Arcade Game. *IEEE Transactions on Computational Intelligence and AI in Games*, 2017.
574. Philipp Rohlfschagen and Simon M. Lucas. Ms Pac-Man versus Ghost team CEC 2011 competition. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 70–77. IEEE, 2011.
575. Edmund T. Rolls. The orbitofrontal cortex and reward. *Cerebral Cortex*, 10(3):284–294, 2000.
576. Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386, 1958.
577. Jonathan Rowe, Bradford Mott, Scott McQuiggan, Jennifer Robison, Sunyoung Lee, and James Lester. Crystal Island: A narrative-centered learning environment for eighth grade microbiology. In *Workshop on Intelligent Educational Games at the 14th International Conference on Artificial Intelligence in Education, Brighton, UK*, pages 11–20, 2009.
578. Jonathan P. Rowe, Lucy R. Shores, Bradford W. Mott, and James C. Lester. Integrating learning, problem solving, and engagement in narrative-centered learning environments. *International Journal of Artificial Intelligence in Education*, 21(1-2):115–133, 2011.
579. David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.
580. Thomas Philip Runarsson and Simon M. Lucas. Coevolution versus self-play temporal difference learning for acquiring position evaluation in small-board Go. *IEEE Transactions on Evolutionary Computation*, 9(6):628–640, 2005.
581. James A. Russell. A circumplex model of affect. *Journal of Personality and Social Psychology*, 39(6):1161, 1980.
582. Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, 1995.
583. Richard M. Ryan, C. Scott Rigby, and Andrew Przybylski. The motivational pull of video games: A self-determination theory approach. *Motivation and emotion*, 30(4):344–360, 2006.
584. Jennifer L. Sabourin and James C. Lester. Affect and engagement in Game-Based Learning environments. *IEEE Transactions on Affective Computing*, 5(1):45–56, 2014.
585. Owen Sacco, Antonios Liapis, and Georgios N. Yannakakis. A holistic approach for semantic-based game generation. In *Computational Intelligence and Games (CIG), 2016 IEEE Conference on*. IEEE, 2016.
586. Frantisek Sailer, Michael Buro, and Marc Lanctot. Adversarial planning through strategy simulation. In *Computational Intelligence and Games, 2007. CIG 2007. IEEE Symposium on*, pages 80–87. IEEE, 2007.
587. Katie Salen and Eric Zimmerman. *Rules of play: Game design fundamentals*. MIT Press, 2004.
588. Christoph Salge, Christian Lipski, Tobias Mahlmann, and Brigitte Mathiak. Using genetically optimized artificial intelligence to improve gameplaying fun for strategical games. In *Sandbox '08: Proceedings of the 2008 ACM SIGGRAPH symposium on Video games*, pages 7–14, New York, NY, USA, 2008. ACM.
589. Spyridon Samothrakis, Simon M. Lucas, Thomas Philip Runarsson, and David Robles. Co-evolving game-playing agents: Measuring performance and intransitivities. *Evolutionary Computation, IEEE Transactions on*, 17(2):213–226, 2013.
590. Spyridon Samothrakis, David Robles, and Simon M. Lucas. Fast approximate max-n Monte Carlo tree search for Ms Pac-Man. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(2):142–154, 2011.
591. Arthur L. Samuel. Some studies in machine learning using the game of Checkers. *IBM Journal of research and development*, 3(3):210–229, 1959.

592. Frederik Schadd, Sander Bakkes, and Pieter Spronck. Opponent modeling in real-time strategy games. In *GAMEON*, pages 61–70, 2007.
593. Jonathan Schaeffer, Neil Burch, Yngvi Björnsson, Akihiro Kishimoto, Martin Müller, Robert Lake, Paul Lu, and Steve Sutphen. Checkers is solved. *Science*, 317(5844):1518–1522, 2007.
594. Jonathan Schaeffer, Robert Lake, Paul Lu, and Martin Bryant. Chinook: the world man-machine Checkers champion. *AI Magazine*, 17(1):21, 1996.
595. Jost Schatzmann, Karl Weilhammer, Matt Stuttle, and Steve Young. A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies. *Knowledge Engineering Review*, 21(2):97–126, 2006.
596. Tom Schaul. A video game description language for model-based or interactive learning. In *Computational Intelligence in Games (CIG), 2013 IEEE Conference on*. IEEE, 2013.
597. Tom Schaul. An extensible description language for video games. *IEEE Transactions on Computational Intelligence and AI in Games*, 6(4):325–331, 2014.
598. Tom Schaul, Julian Togelius, and Jürgen Schmidhuber. Measuring intelligence through games. *arXiv preprint arXiv:1109.1314*, 2011.
599. Jesse Schell. *The Art of Game Design: A book of lenses*. CRC Press, 2014.
600. Klaus R. Scherer. What are emotions? and how can they be measured? *Social Science Information*, 44(4):695–729, 2005.
601. Klaus R. Scherer, Angela Schorr, and Tom Johnstone. *Appraisal processes in emotion: Theory, methods, research*. Oxford University Press, 2001.
602. Jürgen Schmidhuber. Developmental robotics, optimal artificial curiosity, creativity, music, and the fine arts. *Connection Science*, 18(2):173–187, 2006.
603. Jacob Schrum, Igor V. Karpov, and Risto Miikkulainen. UT^2: Human-like behavior via neuroevolution of combat behavior and replay of human traces. In *Computational Intelligence and Games (CIG), 2011 IEEE Conference on*, pages 329–336. IEEE, 2011.
604. Brian Schwab. *AI game engine programming*. Nelson Education, 2009.
605. Brian Schwab, Dave Mark, Kevin Dill, Mike Lewis, and Richard Evans. GDC: Turing tantrums: AI developers rant, 2011.
606. Marco Scirea, Yun-Gyung Cheong, Mark J. Nelson, and Byung-Chull Bae. Evaluating musical foreshadowing of videogame narrative experiences. In *Proceedings of the 9th Audio Mostly: A Conference on Interaction With Sound*. ACM, 2014.
607. Ben Seymour and Samuel M. McClure. Anchors, scales and the relative coding of value in the brain. *Current Opinion in Neurobiology*, 18(2):173–178, 2008.
608. Mohammad Shaker, Mhd Hasan Sarhan, Ola Al Naameh, Noor Shaker, and Julian Togelius. Automatic generation and analysis of physics-based puzzle games. In *Computational Intelligence in Games (CIG), 2013 IEEE Conference on*. IEEE, 2013.
609. Noor Shaker, Stylianos Asteriadis, Georgios N. Yannakakis, and Kostas Karpouzis. A game-based corpus for analysing the interplay between game context and player experience. In *Affective Computing and Intelligent Interaction*, pages 547–556. Springer, 2011.
610. Noor Shaker, Stylianos Asteriadis, Georgios N. Yannakakis, and Kostas Karpouzis. Fusing visual and behavioral cues for modeling user experience in games. *Cybernetics, IEEE Transactions on*, 43(6):1519–1531, 2013.
611. Noor Shaker, Miguel Nicolau, Georgios N. Yannakakis, Julian Togelius, and Michael O’Neil. Evolving levels for Super Mario Bros using grammatical evolution. In *IEEE Conference on Computational Intelligence and Games*, pages 304–311. IEEE, 2012.
612. Noor Shaker, Mohammad Shaker, and Mohamed Abou-Zleikha. Towards generic models of player experience. In *Proceedings, the Eleventh AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*. AAAI Press, 2015.
613. Noor Shaker, Mohammad Shaker, and Julian Togelius. Evolving Playable Content for Cut the Rope through a Simulation-Based Approach. In *AIIDE*, 2013.
614. Noor Shaker, Mohammad Shaker, and Julian Togelius. Ropossum: An Authoring Tool for Designing, Optimizing and Solving Cut the Rope Levels. In *AIIDE*, 2013.
615. Noor Shaker, Gillian Smith, and Georgios N. Yannakakis. Evaluating content generators. In *Procedural Content Generation in Games*, pages 215–224. Springer, 2016.

616. Noor Shaker, Julian Togelius, and Mark J. Nelson, editors. *Procedural Content Generation in Games*. Springer, 2016.
617. Noor Shaker, Julian Togelius, and Georgios N. Yannakakis. Towards Automatic Personalized Content Generation for Platform Games. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*. AAAI Press, October 2010.
618. Noor Shaker, Julian Togelius, and Georgios N. Yannakakis. The experience-driven perspective. In *Procedural Content Generation in Games*, pages 181–194. Springer, 2016.
619. Noor Shaker, Julian Togelius, Georgios N. Yannakakis, Likith Poovanna, Vinay S. Ethiraj, Stefan J. Johansson, Robert G. Reynolds, Leonard K. Heether, Tom Schumann, and Marcus Gallagher. The Turing test track of the 2012 Mario AI championship: entries and evaluation. In *Computational Intelligence in Games (CIG), 2013 IEEE Conference on*. IEEE, 2013.
620. Noor Shaker, Julian Togelius, Georgios N. Yannakakis, Ben Weber, Tomoyuki Shimizu, Tomonori Hashiyama, Nathan Sorenson, Philippe Pasquier, Peter Mawhorter, Glen Takahashi, Gillian Smith, and Robin Baumgarten. The 2010 Mario AI championship: Level generation track. *Computational Intelligence and AI in Games, IEEE Transactions on*, 3(4):332–347, 2011.
621. Noor Shaker, Georgios N. Yannakakis, and Julian Togelius. Crowdsourcing the aesthetics of platform games. *Computational Intelligence and AI in Games, IEEE Transactions on*, 5(3):276–290, 2013.
622. Amirhosein Shantia, Eric Begue, and Marco Wiering. Connectionist reinforcement learning for intelligent unit micro management in StarCraft. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 1794–1801. IEEE, 2011.
623. Manu Sharma, Manish Mehta, Santiago Ontañón, and Ashwin Ram. Player modeling evaluation for interactive fiction. In *Proceedings of the AIIDE 2007 Workshop on Optimizing Player Satisfaction*, pages 19–24, 2007.
624. Nandita Sharma and Tom Gedeon. Objective measures, sensors and computational techniques for stress recognition and classification: A survey. *Computer methods and programs in biomedicine*, 108(3):1287–1301, 2012.
625. Peter Shizgal and Andreas Arvanitogiannis. Gambling on dopamine. *Science*, 299(5614):1856–1858, 2003.
626. Yoav Shoham and Kevin Leyton-Brown. *Multiagent systems: Algorithmic, game-theoretic, and logical foundations*. Cambridge University Press, 2008.
627. Alexander Shoulson, Francisco M. Garcia, Matthew Jones, Robert Mead, and Norman I. Badler. Parameterizing behavior trees. In *International Conference on Motion in Games*, pages 144–155. Springer, 2011.
628. Nikolaos Sidorakis, George Alex Koulieris, and Katerina Mania. Binocular eye-tracking for the control of a 3D immersive multimedia user interface. In *Everyday Virtual Reality (WEVR), 2015 IEEE 1st Workshop on*, pages 15–18. IEEE, 2015.
629. David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.
630. Herbert A. Simon. A behavioral model of rational choice. *The quarterly journal of economics*, 69(1):99–118, 1955.
631. Shawn Singh, Mubbasir Kapadia, Glenn Reinman, and Petros Faloutsos. Footstep navigation for dynamic crowds. *Computer Animation and Virtual Worlds*, 22(2-3):151–158, 2011.
632. Moshe Sipper. *Evolved to Win*. Lulu.com, 2011.
633. Burrhus Frederic Skinner. *The behavior of organisms: An experimental analysis*. BF Skinner Foundation, 1990.
634. Ruben M. Smelik, Tim Tutenel, Klaas Jan de Kraker, and Rafael Bidarra. Interactive creation of virtual worlds using procedural sketching. In *Proceedings of Eurographics*, 2010.
635. Adam M. Smith, Erik Andersen, Michael Mateas, and Zoran Popović. A case study of expressively constrainable level design automation tools for a puzzle game. In *Proceedings of the International Conference on the Foundations of Digital Games*, pages 156–163. ACM, 2012.

636. Adam M. Smith, Chris Lewis, Kenneth Hullett, Gillian Smith, and Anne Sullivan. An inclusive taxonomy of player modeling. Technical Report UCSC-SOE-11-13, University of California, Santa Cruz, 2011.
637. Adam M. Smith and Michael Mateas. Variations forever: Flexibly generating rulesets from a sculptable design space of mini-games. In *Computational Intelligence and Games (CIG), 2010 IEEE Symposium on*, pages 273–280. IEEE, 2010.
638. Adam M. Smith and Michael Mateas. Answer set programming for procedural content generation: A design space approach. *Computational Intelligence and AI in Games, IEEE Transactions on*, 3(3):187–200, 2011.
639. Adam M. Smith, Mark J. Nelson, and Michael Mateas. Ludocore: A logical game engine for modeling videogames. In *Computational Intelligence and Games (CIG), 2010 IEEE Symposium on*, pages 91–98. IEEE, 2010.
640. Gillian Smith and Jim Whitehead. Analyzing the expressive range of a level generator. In *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*. ACM, 2010.
641. Gillian Smith, Jim Whitehead, and Michael Mateas. Tanagra: A mixed-initiative level design tool. In *Proceedings of the Fifth International Conference on the Foundations of Digital Games*, pages 209–216. ACM, 2010.
642. Gillian Smith, Jim Whitehead, and Michael Mateas. Tanagra: Reactive planning and constraint solving for mixed-initiative level design. *Computational Intelligence and AI in Games, IEEE Transactions on*, 3(3):201–215, 2011.
643. Ian Sneddon, Gary McKeown, Margaret McRorie, and Tijana Vukicevic. Cross-cultural patterns in dynamic ratings of positive and negative natural emotional behaviour. *PloS ONE*, 6(2), 2011.
644. Sam Snodgrass and Santiago Ontañón. A Hierarchical MdMC Approach to 2D Video Game Map Generation. In *Eleventh Artificial Intelligence and Interactive Digital Entertainment Conference*, 2015.
645. Dennis Soemers. Tactical planning using MCTS in the game of StarCraft, 2014. Bachelor Thesis, Department of Knowledge Engineering, Maastricht University.
646. Andreas Sonderegger, Andreas Uebelbacher, Manuela Pugliese, and Juergen Sauer. The influence of aesthetics in usability testing: the case of dual-domain products. In *Proceedings of the Conference on Human Factors in Computing Systems*, pages 21–30, 2014.
647. Bhuman Soni and Philip Hingston. Bots trained to play like a human are more fun. In *IEEE International Joint Conference on Neural Networks (IJCNN); IEEE World Congress on Computational Intelligence*, pages 363–369. IEEE, 2008.
648. Patrik D. Sørensen, Jeppeh M. Olsen, and Sebastian Risi. Interactive Super Mario Bros Evolution. In *Proceedings of the 2016 Genetic and Evolutionary Computation Conference*, pages 41–42. ACM, 2016.
649. Nathan Sorenson and Philippe Pasquier. Towards a generic framework for automated video game level creation. *Applications of Evolutionary Computation*, pages 131–140, 2010.
650. Pieter Spronck, Marc Ponsen, Ida Sprinkhuizen-Kuyper, and Eric Postma. Adaptive game AI with dynamic scripting. *Machine Learning*, 63(3):217–248, 2006.
651. Pieter Spronck, Ida Sprinkhuizen-Kuyper, and Eric Postma. Difficulty scaling of game AI. In *Proceedings of the 5th International Conference on Intelligent Games and Simulation (GAME-ON 2004)*, pages 33–37, 2004.
652. Ramakrishnan Srikanth and Rakesh Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *International Conference on Extending Database Technology*, pages 1–17. Springer, 1996.
653. Kenneth O. Stanley. Compositional Pattern Producing Networks: A novel abstraction of development. *Genetic Programming and Evolvable Machines*, 8(2):131–162, 2007.
654. Kenneth O. Stanley, Bobby D. Bryant, and Risto Miikkulainen. Real-time neuroevolution in the NERO video game. *Evolutionary Computation, IEEE Transactions on*, 9(6):653–668, 2005.
655. Kenneth O. Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127, 2002.

656. Kenneth O. Stanley and Risto Miikkulainen. Evolving a roving eye for Go. In *Genetic and Evolutionary Computation Conference*, pages 1226–1238. Springer, 2004.
657. Stanley Smith Stevens. On the Theory of Scales of Measurement. *Science*, 103(2684):677–680, 1946.
658. Neil Stewart, Gordon D. A. Brown, and Nick Chater. Absolute identification by relative judgment. *Psychological Review*, 112(4):881, 2005.
659. Andreas Stieglar, Keshav Dahal, Johannes Maucher, and Daniel Livingstone. Symbolic Reasoning for Hearthstone. *IEEE Transactions on Computational Intelligence and AI in Games*, 2017.
660. Jeff Stuckman and Guo-Qiang Zhang. Mastermind is NP-complete. *arXiv preprint cs/0512049*, 2005.
661. Nathan Sturtevant. Memory-Efficient Pathfinding Abstractions. In *AI Programming Wisdom 4*. Charles River Media, 2008.
662. Nathan Sturtevant and Steve Rabin. Canonical orderings on grids. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 683–689, 2016.
663. Nathan R. Sturtevant. Benchmarks for grid-based pathfinding. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(2):144–148, 2012.
664. Nathan R. Sturtevant and Richard E. Korf. On pruning techniques for multi-player games. *Proceedings of The National Conference on Artificial Intelligence (AAAI)*, pages 201–208, 2000.
665. Nathan R. Sturtevant, Jason Traish, James Tulip, Tansel Uras, Sven Koenig, Ben Strasser, Adi Botea, Daniel Harabor, and Steve Rabin. The Grid-Based Path Planning Competition: 2014 Entries and Results. In *Eighth Annual Symposium on Combinatorial Search*, pages 241–251, 2015.
666. Adam James Summerville and Michael Mateas. Mystical Tutor: A Magic: The Gathering Design Assistant via Denoising Sequence-to-Sequence Learning. In *Twelfth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2016.
667. Adam James Summerville, Shweta Philip, and Michael Mateas. MCMCTS PCG 4 SMB: Monte Carlo Tree Search to Guide Platformer Level Generation. In *Eleventh Artificial Intelligence and Interactive Digital Entertainment Conference*, 2015.
668. Adam James Summerville, Sam Snodgrass, Matthew Guzdial, Christoffer Holmgård, Amy K. Hoover, Aaron Isaksen, Andy Nealen, and Julian Togelius. Procedural Content Generation via Machine Learning (PCGML). *arXiv preprint arXiv:1702.00539*, 2017.
669. Adam James Summerville, Sam Snodgrass, Michael Mateas, and Santiago Ontañón Villar. The VGLC: The Video Game Level Corpus. *arXiv preprint arXiv:1606.07487*, 2016.
670. Petra Sundström. *Exploring the affective loop*. PhD thesis, Stockholm University, 2005.
671. Ben Sunshine-Hill, Michael Robbins, and Chris Jurney. Off the Beaten Path: Non-Traditional Uses of AI. In *Game Developers Conference, AI Summit*, 2012.
672. Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: An introduction*. MIT Press, 1998.
673. Reid Swanson and Andrew S. Gordon. Say anything: Using textual case-based reasoning to enable open-domain interactive storytelling. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 2(3):16, 2012.
674. William R. Swartout, Jonathan Gratch, Randall W. Hill Jr, Eduard Hovy, Stacy Marsella, Jeff Rickel, and David Traum. Toward virtual humans. *AI Magazine*, 27(2):96, 2006.
675. Penelope Sweetser, Daniel M. Johnson, and Peta Wyeth. Revisiting the GameFlow model with detailed heuristics. *Journal: Creative Technologies*, 2012(3), 2012.
676. Penelope Sweetser and Janet Wiles. Scripting versus emergence: issues for game developers and players in game environment design. *International Journal of Intelligent Games and Simulations*, 4(1):1–9, 2005.
677. Penelope Sweetser and Janet Wiles. Using cellular automata to facilitate emergence in game environments. In *Proceedings of the 4th International Conference on Entertainment Computing (ICEC05)*, 2005.
678. Penelope Sweetser and Peta Wyeth. GameFlow: a model for evaluating player enjoyment in games. *Computers in Entertainment (CIE)*, 3(3):3–3, 2005.

679. Maciej Świechowski and Jacek Mańdziuk. Self-adaptation of playing strategies in general game playing. *IEEE Transactions on Computational Intelligence and AI in Games*, 6(4):367–381, 2014.
680. Gabriel Synnaeve and Pierre Bessière. Multiscale Bayesian Modeling for RTS Games: An Application to StarCraft AI. *IEEE Transactions on Computational intelligence and AI in Games*, 8(4):338–350, 2016.
681. Gabriel Synnaeve, Nantas Nardelli, Alex Auvolat, Soumith Chintala, Timothée Lacroix, Zeming Lin, Florian Richoux, and Nicolas Usunier. TorchCraft: a Library for Machine Learning Research on Real-Time Strategy Games. *arXiv preprint arXiv:1611.00625*, 2016.
682. Nicolas Szilas. IDtension: a narrative engine for Interactive Drama. In *Proceedings of the Technologies for Interactive Digital Storytelling and Entertainment (TIDSE) Conference*, pages 1–11, 2003.
683. Niels A. Taatgen, Marcia van Oploo, Jos Braaksma, and Jelle Niemantsverdriet. How to construct a believable opponent using cognitive modeling in the game of set. In *Proceedings of the Fifth International Conference on Cognitive Modeling*, pages 201–206, 2003.
684. Nima Taghipour, Ahmad Kardan, and Saeed Shiry Ghidary. Usage-based web recommendations: a reinforcement learning approach. In *Proceedings of the 2007 ACM Conference on Recommender Systems*, pages 113–120. ACM, 2007.
685. Bulent Tastan and Gita Reese Sukthankar. Learning policies for first person shooter games using inverse reinforcement learning. In *Seventh Artificial Intelligence and Interactive Digital Entertainment Conference*, 2011.
686. Shoshannah Tekofsky, Pieter Spronck, Aske Plaat, Jaap van Den Herik, and Jan Broersen. Play style: Showing your age. In *Computational Intelligence in Games (CIG), 2013 IEEE Conference on*. IEEE, 2013.
687. Shoshannah Tekofsky, Pieter Spronck, Aske Plaat, Jaap van den Herik, and Jan Broersen. Psyops: Personality assessment through gaming behavior. In *BNAIC 2013: Proceedings of the 25th Benelux Conference on Artificial Intelligence, Delft, The Netherlands, November 7-8, 2013*, 2013.
688. Gerald Tesauro. Practical issues in temporal difference learning. *Machine learning*, 8(3-4):257–277, 1992.
689. Gerald Tesauro. Temporal difference learning and TD-Gammon. *Communications of the ACM*, 38(3):58–68, 1995.
690. Ruck Thawonmas, Yoshitaka Kashifiji, and Kuan-Ta Chen. Detection of MMORPG bots based on behavior analysis. In *Proceedings of the 2008 International Conference on Advances in Computer Entertainment Technology*, pages 91–94. ACM, 2008.
691. Michael Thielscher. A General Game Description Language for Incomplete Information Games. In *AAAI*, pages 994–999, 2010.
692. William R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.
693. David Thue, Vadim Bulitko, Marcia Spetch, and Eric Wasylisen. Interactive Storytelling: A Player Modelling Approach. In *AIIDE*, pages 43–48, 2007.
694. Christian Thurau, Christian Bauckhage, and Gerhard Sagerer. Learning human-like opponent behavior for interactive computer games. *Pattern Recognition, Lecture Notes in Computer Science 2781*, pages 148–155, 2003.
695. Christian Thurau, Christian Bauckhage, and Gerhard Sagerer. Imitation learning at all levels of game AI. In *Proceedings of the International Conference on Computer Games, Artificial Intelligence, Design and Education*, 2004.
696. Christian Thurau, Christian Bauckhage, and Gerhard Sagerer. Learning human-like Movement Behavior for Computer Games. In S. Schaal, A. Ijspeert, A. Billard, S. Vijayakumar, J. Hallam, and J.-A. Meyer, editors, *From Animals to Animats 8: Proceedings of the Eighth International Conference on Simulation of Adaptive Behavior (SAB-04)*, pages 315–323, Santa Monica, CA, July 2004. The MIT Press.
697. Tim J. W. Tijs, Dirk Brokken, and Wijnand A. Ijsselsteijn. Dynamic game balancing by recognizing affect. In *Proceedings of International Conference on Fun and Games*, pages 88–93. Springer, 2008.

698. Julian Togelius. Evolution of a subsumption architecture neurocontroller. *Journal of Intelligent & Fuzzy Systems*, 15(1):15–20, 2004.
699. Julian Togelius. A procedural critique of deontological reasoning. In *Proceedings of DiGRA*, 2011.
700. Julian Togelius. AI researchers, Video Games are your friends! In *Computational Intelligence*, pages 3–18. Springer, 2015.
701. Julian Togelius. How to run a successful game-based AI competition. *IEEE Transactions on Computational Intelligence and AI in Games*, 8(1):95–100, 2016.
702. Julian Togelius, Alex J. Champandard, Pier Luca Lanzi, Michael Mateas, Ana Paiva, Mike Preuss, and Kenneth O. Stanley. Procedural Content Generation in Games: Goals, Challenges and Actionable Steps. *Dagstuhl Follow-Ups*, 6, 2013.
703. Julian Togelius, Renzo De Nardi, and Simon M. Lucas. Making racing fun through player modeling and track evolution. In *Proceedings of the SAB’06 Workshop on Adaptive Approaches for Optimizing Player Satisfaction in Computer and Physical Games*, 2006.
704. Julian Togelius, Renzo De Nardi, and Simon M. Lucas. Towards automatic personalised content creation for racing games. In *Computational Intelligence and Games, 2007. CIG 2007. IEEE Symposium on*, pages 252–259. IEEE, 2007.
705. Julian Togelius, Sergey Karakovskiy, and Robin Baumgarten. The 2009 Mario AI competition. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*. IEEE, 2010.
706. Julian Togelius, Sergey Karakovskiy, Jan Koutrník, and Jürgen Schmidhuber. Super Mario evolution. In *Computational Intelligence and Games, 2009. CIG 2009. IEEE Symposium on*, pages 156–161. IEEE, 2009.
707. Julian Togelius and Simon M. Lucas. Evolving controllers for simulated car racing. In *IEEE Congress on Evolutionary Computation*, pages 1906–1913. IEEE, 2005.
708. Julian Togelius and Simon M. Lucas. Arms races and car races. In *Parallel Problem Solving from Nature-PPSN IX*, pages 613–622. Springer, 2006.
709. Julian Togelius and Simon M. Lucas. Evolving robust and specialized car racing skills. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 1187–1194. IEEE, 2006.
710. Julian Togelius, Simon M. Lucas, Ho Duc Thang, Jonathan M. Garibaldi, Tomoharu Nakashima, Chin Hiong Tan, Itamar Elhanany, Shay Berant, Philip Hingston, Robert M. MacCallum, Thomas Haferlach, Aravind Gowrisankar, and Pete Burrow. The 2007 IEEE CEC Simulated Car Racing Competition. *Genetic Programming and Evolvable Machines*, 9(4):295–329, 2008.
711. Julian Togelius, Mark J. Nelson, and Antonios Liapis. Characteristics of generatable games. In *Proceedings of the Fifth Workshop on Procedural Content Generation in Games*, 2014.
712. Julian Togelius, Mike Preuss, Nicola Beume, Simon Wessing, Johan Hagelbäck, and Georgios N. Yannakakis. Multiobjective exploration of the StarCraft map space. In *Computational Intelligence and Games (CIG), 2010 IEEE Symposium on*, pages 265–272. IEEE, 2010.
713. Julian Togelius, Mike Preuss, and Georgios N. Yannakakis. Towards multiobjective procedural map generation. In *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*. ACM, 2010.
714. Julian Togelius, Tom Schaul, Jürgen Schmidhuber, and Faustino Gomez. Countering poisonous inputs with memetic neuroevolution. In *International Conference on Parallel Problem Solving from Nature*, pages 610–619. Springer, 2008.
715. Julian Togelius, Tom Schaul, Daan Wierstra, Christian Igel, Faustino Gomez, and Jürgen Schmidhuber. Ontogenetic and phylogenetic reinforcement learning. *Künstliche Intelligenz*, 23(3):30–33, 2009.
716. Julian Togelius and Jürgen Schmidhuber. An experiment in automatic game design. In *Computational Intelligence and Games, 2008. CIG’08. IEEE Symposium On*, pages 111–118. IEEE, 2008.
717. Julian Togelius, Noor Shaker, Sergey Karakovskiy, and Georgios N. Yannakakis. The Mario AI championship 2009–2012. *AI Magazine*, 34(3):89–92, 2013.
718. Julian Togelius and Georgios N. Yannakakis. General General Game AI. In *2016 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 2016.

719. Julian Togelius, Georgios N. Yannakakis, Sergey Karakovskiy, and Noor Shaker. Assessing believability. In Philip Hingston, editor, *Believable bots*, pages 215–230. Springer, 2012.
720. Julian Togelius, Georgios N. Yannakakis, Kenneth O. Stanley, and Cameron Browne. Search-based procedural content generation: A taxonomy and survey. *Computational Intelligence and AI in Games, IEEE Transactions on*, 3(3):172–186, 2011.
721. Simone Tognetti, Maurizio Garbarino, Andrea Bonarini, and Matteo Matteucci. Modeling enjoyment preference from physiological responses in a car racing game. In *Computational Intelligence and Games (CIG), 2010 IEEE Symposium on*, pages 321–328. IEEE, 2010.
722. Paul Tozour and I. S. Austin. Building a near-optimal navigation mesh. *AI Game Programming Wisdom*, 1:298–304, 2002.
723. Mike Treanor, Bryan Blackford, Michael Mateas, and Ian Bogost. Game-O-Matic: Generating Videogames that Represent Ideas. In *Procedural Content Generation Workshop at the Foundations of Digital Games Conference*. ACM, 2012.
724. Mike Treanor, Alexander Zook, Mirjam P. Eladhar, Julian Togelius, Gillian Smith, Michael Cook, Tommy Thompson, Brian Magerko, John Levine, and Adam Smith. AI-based game design patterns. 2015.
725. Alan M. Turing. Digital computers applied to games. *Faster than thought*, 101, 1953.
726. Hiroto Udagawa, Tarun Narasimhan, and Shim-Young Lee. Fighting Zombies in Minecraft With Deep Reinforcement Learning. Technical report, Stanford University, 2016.
727. Alfred Ultsch. Data mining and knowledge discovery with emergent self-organizing feature maps for multivariate time series. *Kohonen Maps*, 46:33–46, 1999.
728. Alberto Uriarte and Santiago Ontañón. Automatic learning of combat models for RTS games. In *Eleventh Artificial Intelligence and Interactive Digital Entertainment Conference*, 2015.
729. Nicolas Usunier, Gabriel Synnaeve, Zeming Lin, and Soumith Chintala. Episodic Exploration for Deep Deterministic Policies: An Application to StarCraft Micromanagement Tasks. *arXiv preprint arXiv:1609.02993*, 2016.
730. Josep Valls-Vargas, Santiago Ontañón, and Jichen Zhu. Towards story-based content generation: From plot-points to maps. In *Computational Intelligence in Games (CIG), 2013 IEEE Conference on*. IEEE, 2013.
731. Wouter van den Hoogen, Wijnand A. IJsselsteijn, and Yvonne de Kort. Exploring behavioral expressions of player experience in digital games. In *Proceedings of the Workshop on Facial and Bodily Expression for Control and Adaptation of Games (ECAG)*, pages 11–19, 2008.
732. Roland van der Linden, Ricardo Lopes, and Rafael Bidarra. Procedural generation of dungeons. *Computational Intelligence and AI in Games, IEEE Transactions on*, 6(1):78–89, 2014.
733. Pascal van Hentenryck. *Constraint satisfaction in logic programming*. MIT Press, Cambridge, 1989.
734. Niels van Hoorn, Julian Togelius, and Jürgen Schmidhuber. Hierarchical controller learning in a first-person shooter. In *Computational Intelligence and Games, 2009. CIG 2009. IEEE Symposium on*, pages 294–301. IEEE, 2009.
735. Niels van Hoorn, Julian Togelius, Daan Wierstra, and Jürgen Schmidhuber. Robust player imitation using multiobjective evolution. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 652–659. IEEE, 2009.
736. Giel van Lankveld, Sonny Schreurs, Pieter Spronck, and Jaap van Den Herik. Extraversion in games. In *International Conference on Computers and Games*, pages 263–275. Springer, 2010.
737. Giel van Lankveld, Pieter Spronck, Jaap van den Herik, and Arnoud Arntz. Games as personality profiling tools. In *Computational Intelligence and Games (CIG), 2011 IEEE Conference on*, pages 197–202. IEEE, 2011.
738. Harm van Seijen, Mehdi Fatemi, Joshua Romoff, Romain Laroche, Tavian Barnes, and Jeffrey Tsang. Hybrid Reward Architecture for Reinforcement Learning. *arXiv preprint arXiv:1706.04208*, 2017.
739. Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, pages 1096–1103. ACM, 2008.

740. Madhubalan Viswanathan. Measurement of individual differences in preference for numerical information. *Journal of Applied Psychology*, 78(5):741–752, 1993.
741. Thurid Vogt and Elisabeth André. Comparing feature sets for acted and spontaneous speech in view of automatic emotion recognition. In *Proceedings of IEEE International Conference on Multimedia and Expo (ICME)*, pages 474–477. IEEE, 2005.
742. John Von Neumann. The general and logical theory of automata. *Cerebral Mechanisms in Behavior*, 1(41):1–2, 1951.
743. John Von Neumann and Oskar Morgenstern. *Theory of games and economic behavior*. Princeton University Press, 1944.
744. Karol Walédzik and Jacek Mańdziuk. An automatically generated evaluation function in general game playing. *IEEE Transactions on Computational Intelligence and AI in Games*, 6(3):258–270, 2014.
745. Che Wang, Pan Chen, Yuanda Li, Christoffer Holmgård, and Julian Togelius. Portfolio Online Evolution in StarCraft. In *Twelfth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2016.
746. Colin D. Ward and Peter I. Cowling. Monte Carlo search applied to card selection in Magic: The Gathering. In *IEEE Symposium on Computational Intelligence and Games (CIG)*, pages 9–16. IEEE, 2009.
747. Joe H. Ward Jr. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58(301):236–244, 1963.
748. Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3-4):279–292, 1992.
749. Ben G. Weber. ABL versus Behavior Trees. *Gamasutra*, 2012.
750. Ben G. Weber and Michael Mateas. A data mining approach to strategy prediction. In *2009 IEEE Symposium on Computational Intelligence and Games*, pages 140–147. IEEE, 2009.
751. Joseph Weizenbaum. ELIZA—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45, 1966.
752. Paul John Werbos. *Beyond regression: new tools for prediction and analysis in the behavioral sciences*. PhD thesis, Harvard University, 1974.
753. Daan Wierstra, Tom Schaul, Jan Peters, and Juergen Schmidhuber. Natural evolution strategies. In *IEEE Congress on Evolutionary Computation (CEC) 2008. (IEEE World Congress on Computational Intelligence)*, pages 3381–3387. IEEE, 2008.
754. Geraint A. Wiggins. A preliminary framework for description, analysis and comparison of creative systems. *Knowledge-Based Systems*, 19(7):449–458, 2006.
755. Minecraft Wiki. Minecraft. Mojang AB, Stockholm, Sweden, 2013.
756. David H. Wolpert and William G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.
757. Robert F. Woodbury. Searching for designs: Paradigm and practice. *Building and Environment*, 26(1):61–73, 1991.
758. Steven Woodcock. Game AI: The State of the Industry 2000-2001: It's not Just Art, It's Engineering. *Game Developer Magazine*, 2001.
759. Xindong Wu, Vipin Kumar, J. Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J. McLachlan, Angus Ng, Bing Liu, S. Yu Philip, Zhi-Hua Zhou, Michael Steinbach, David J. Hand, and Dan Steinberg. Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14(1):1–37, 2008.
760. Kaito Yamamoto, Syunsuke Mizuno, Chun Yin Chu, and Ruck Thawonmas. Deduction of fighting-game countermeasures using the k-nearest neighbor algorithm and a game simulator. In *Computational Intelligence and Games (CIG), 2014 IEEE Conference on*. IEEE, 2014.
761. Yi-Hsuan Yang and Homer H. Chen. Ranking-based emotion recognition for music organization and retrieval. *Audio, Speech, and Language Processing, IEEE Transactions on*, 19(4):762–774, 2011.
762. Georgios N. Yannakakis. *AI in Computer Games: Generating Interesting Interactive Opponents by the use of Evolutionary Computation*. PhD thesis, University of Edinburgh, November 2005.

763. Georgios N. Yannakakis. Preference learning for affective modeling. In *Affective Computing and Intelligent Interaction and Workshops, 2009. ACII 2009. 3rd International Conference on*, pages 1–6. IEEE, 2009.
764. Georgios N. Yannakakis. Game AI revisited. In *Proceedings of the 9th conference on Computing Frontiers*, pages 285–292. ACM, 2012.
765. Georgios N. Yannakakis, Roddy Cowie, and Carlos Busso. The Ordinal Nature of Emotions. In *Affective Computing and Intelligent Interaction (ACII), 2017 International Conference on*, 2017.
766. Georgios N. Yannakakis and John Hallam. Evolving Opponents for Interesting Interactive Computer Games. In S. Schaal, A. Ijspeert, A. Billard, S. Vijayakumar, J. Hallam, and J.-A. Meyer, editors, *From Animals to Animats 8: Proceedings of the 8<sup>th</sup> International Conference on Simulation of Adaptive Behavior (SAB-04)*, pages 499–508, Santa Monica, CA, July 2004. The MIT Press.
767. Georgios N. Yannakakis and John Hallam. A Generic Approach for Generating Interesting Interactive Pac-Man Opponents. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games*, 2005.
768. Georgios N. Yannakakis and John Hallam. A generic approach for obtaining higher entertainment in predator/prey computer games. *Journal of Game Development*, 1(3):23–50, December 2005.
769. Georgios N. Yannakakis and John Hallam. Modeling and augmenting game entertainment through challenge and curiosity. *International Journal on Artificial Intelligence Tools*, 16(06):981–999, 2007.
770. Georgios N. Yannakakis and John Hallam. Towards optimizing entertainment in computer games. *Applied Artificial Intelligence*, 21(10):933–971, 2007.
771. Georgios N. Yannakakis and John Hallam. Entertainment modeling through physiology in physical play. *International Journal of Human-Computer Studies*, 66(10):741–755, 2008.
772. Georgios N. Yannakakis and John Hallam. Real-time game adaptation for optimizing player satisfaction. *IEEE Transactions on Computational Intelligence and AI in Games*, 1(2):121–133, 2009.
773. Georgios N. Yannakakis and John Hallam. Rating vs. preference: A comparative study of self-reporting. In *Affective Computing and Intelligent Interaction*, pages 437–446. Springer, 2011.
774. Georgios N. Yannakakis, Antonios Liapis, and Constantine Alexopoulos. Mixed-initiative co-creativity. In *Proceedings of the 9th Conference on the Foundations of Digital Games*, 2014.
775. Georgios N. Yannakakis, Henrik Hautop Lund, and John Hallam. Modeling children’s entertainment in the playware playground. In *2006 IEEE Symposium on Computational Intelligence and Games*, pages 134–141. IEEE, 2006.
776. Georgios N. Yannakakis and Manolis Maragoudakis. Player modeling impact on player’s entertainment in computer games. In *Proceedings of International Conference on User Modeling (UM)*. Springer, 2005.
777. Georgios N. Yannakakis and Héctor P. Martínez. Grounding truth via ordinal annotation. In *Affective Computing and Intelligent Interaction (ACII), 2015 International Conference on*, pages 574–580. IEEE, 2015.
778. Georgios N. Yannakakis and Héctor P. Martínez. Ratings are Overrated! *Frontiers in ICT*, 2:13, 2015.
779. Georgios N. Yannakakis, Héctor P. Martínez, and Maurizio Garbarino. Psychophysiology in games. In *Emotion in Games: Theory and Praxis*, pages 119–137. Springer, 2016.
780. Georgios N. Yannakakis, Héctor P. Martínez, and Arnav Jhala. Towards affective camera control in games. *User Modeling and User-Adapted Interaction*, 20(4):313–340, 2010.
781. Georgios N. Yannakakis and Ana Paiva. Emotion in games. *Handbook on Affective Computing*, pages 459–471, 2014.
782. Georgios N. Yannakakis, Pieter Spronck, Daniele Loiacono, and Elisabeth André. Player modeling. *Dagstuhl Follow-Ups*, 6, 2013.

783. Georgios N. Yannakakis and Julian Togelius. Experience-driven procedural content generation. *Affective Computing, IEEE Transactions on*, 2(3):147–161, 2011.
784. Georgios N. Yannakakis and Julian Togelius. Experience-driven procedural content generation. In *Affective Computing and Intelligent Interaction (ACII), 2015 International Conference on*, pages 519–525. IEEE, 2015.
785. Georgios N. Yannakakis and Julian Togelius. A panorama of artificial and computational intelligence in games. *IEEE Transactions on Computational Intelligence and AI in Games*, 7(4):317–335, 2015.
786. Xin Yao. Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423–1447, 1999.
787. Nick Yee. The demographics, motivations, and derived experiences of users of massively multi-user online graphical environments. *Presence: Teleoperators and virtual environments*, 15(3):309–329, 2006.
788. Nick Yee, Nicolas Ducheneaut, Les Nelson, and Peter Likarish. Introverted elves & conscientious gnomes: the expression of personality in World of Warcraft. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 753–762. ACM, 2011.
789. Serdar Yildirim, Shrikanth Narayanan, and Alexandros Potamianos. Detecting emotional state of a child in a conversational computer game. *Computer Speech & Language*, 25(1):29–44, 2011.
790. Shubu Yoshida, Makoto Ishihara, Taichi Miyazaki, Yuto Nakagawa, Tomohiro Harada, and Ruck Thawonmas. Application of Monte-Carlo tree search in a fighting game AI. In *Consumer Electronics, 2016 IEEE 5th Global Conference on*. IEEE, 2016.
791. David Young. *Learning game AI programming with Lua*. Packt Publishing Ltd, 2014.
792. R. Michael Young, Mark O. Riedl, Mark Branly, Arnav Jhala, R. J. Martin, and C. J. Saretto. An architecture for integrating plan-based behavior generation with interactive game environments. *Journal of Game Development*, 1(1):51–70, 2004.
793. Mohammed J. Zaki. SPADE: An efficient algorithm for mining frequent sequences. *Machine Learning*, 42(1-2):31–60, 2001.
794. Zhihong Zeng, Maja Pantic, Glenn I. Roisman, and Thomas S. Huang. A survey of affect recognition methods: Audio, visual, and spontaneous expressions. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(1):39–58, 2009.
795. Jiakai Zhang and Kyunghyun Cho. Query-efficient imitation learning for end-to-end autonomous driving. *arXiv preprint arXiv:1605.06450*, 2016.
796. Peng Zhang and Jochen Renz. Qualitative Spatial Representation and Reasoning in Angry Birds: The Extended Rectangle Algebra. In *Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning*, 2014.
797. Martin Zinkevich, Michael Johanson, Michael Bowling, and Carmelo Piccione. Regret minimization in games with incomplete information. In *Advances in Neural Information Processing Systems*, pages 1729–1736, 2008.
798. Albert L. Zobrist. *Feature extraction and representation for pattern recognition and the game of Go*. PhD thesis, The University of Wisconsin, Madison, 1970.
799. Alexander Zook. Game AGI beyond Characters. In *Integrating Cognitive Architectures into Virtual Character Design*, pages 266–293. IGI Global, 2016.
800. Alexander Zook and Mark O. Riedl. A Temporal Data-Driven Player Model for Dynamic Difficulty Adjustment. In *8th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*. AAAI, 2012.
801. Robert Zubek and Ian Horswill. Hierarchical Parallel Markov Models of Interaction. In *AIIDE*, pages 141–146, 2005.



# Index

- $\varepsilon$ -greedy, 46, 73
- $n$ -grams, 173, 190
- 2K BotPrize, 96, **139**, 267, 274
- A Behavior Language, 35
- AAA games, 152, 217, 270
- Adaptive games, 153, 284
- Adversarial games, 99, 113
- Adversarial planning, 119, 137
- Adversarial player, 205
- Adversarial search, 43
- Affect expression, 284
- Affect modeling, 284
- Affective computing, 21, 143, 203, 228, 284
  - Context, 284
  - Affective interaction, 217
  - Affective loop, 18, 22, 284
  - Affective sciences, 208
  - Agent architecture, 143, 274
  - AI researcher, 3, 6, 263, 264
  - AI-assisted game design, 176, 272
  - AI-based game design, 286
  - AI-complete, 159
  - ALE, *see* Arcade Learning Environment
  - Analytics, 206
  - Angry Birds AI competition, 146
  - ANN, *see* Artificial neural networks
  - ANN function approximator, 84, 87, 116
- Annotation, 200, 219
  - Absolute, **228**
  - Continuous, **222**
  - Discrete, **222**
  - During-experience, **225**
  - First-person, **221**
  - Forced, 221
  - Free-response, 221
  - Ordinal, 228, 233
- Post-experience, **225**
- Pre-experience, **224**
- Relative, **228**
- Third-person, **221**
- Time-continuous, **222**
- Time-discrete, **223**
- Answer set programming, 146, 162, 178, 193, 198
- Anytime algorithm, 45, 161
- Arcade games, 123
- Arcade learning environment, 23, 101, **129**, 281
- Arcade video games, 23
- Architecture, 157, 184
- Artificial general intelligence, 266, 271, 290
- Artificial neural networks, 31, **59**, 65, 174, 231
  - Activation function, 59, 60
  - Backpropagation, 30, 60, 63
  - Deep architectures, 65
  - Learning rate, 63
  - Multi-layer perceptron, 60
  - Neuron, 59
  - ReLU, 60
- ASP, *see* Answer set programming
- Atari 2600, 101, 106, 116, 124, 129
- Audio designer, 187
- Autoencoders, 65, 141, 178, 199, 282
  - Stacked autoencoders, 285
  - Variational autoencoders, 171
- Automatic game testing, 279
- Avatar, 203
- Backgammon, 45, 85, 116, 120
- Balance, 192
- Behavior trees, 31, **34**
  - Decorator, 35
  - Selector, 35

- Sequence, 35
- Behavioral economics, 228
- Believability, 190, 268
- Believable agents, 144, 190, 207
- Believable NPC, 143
- Benchmarks, 101, 266, 281
- Best-first search, **41**
  - A\*, 41, 100, 110, 127
  - Uniform-cost grids, 41
- Biofeedback, 160, 217, 273
- Board games, 3, 8, 10, 23, **119**, 192, 266, 276, 279
- Bodily expressions, 215
- Body movement, 18
- Body stance, 18
- Branching factor, 45, 103
- BT, *see* Behavior trees
- Card games, **121**
- Casual games, **145**
- Cellular automata, 156, 165
- Censorship, 289
- Cheating, 95, 106, 108
- Checkers, 8, 43, 120
  - Chinook, 8
- Chess, 8, 43, 107, 119
  - Deep Blue, 8, 16, 43, 93
- CI, *see* Computational intelligence
- Classes, 226
  - Class splitting criteria, 228
- Classification, 58, 231, 234, 243
- Clustering, **77**, 80, 207
  - Hierarchical, 79
  - k-means, **78**
- Co-creation, 177
- Co-creativity, 154
- Collaborative filtering, 289
- Commodore 64, 124
- Compositional pattern producing networks, 60, 84
- Computational creativity, 22, 180, 270, 286
  - Exploratory, 283
  - Transformational, 283
- Computational intelligence, 7
- Computational narrative, 189
- Concept map, 195
- ConceptNet, 195
- Constraint satisfaction, 198
- Constraint solvers, 161
  - SAT, 161
- Constraints, 198
- Content, 19, 151, 154
  - Audio, 19, **187**
  - Levels, **184**
- Maps, **184**
- Mechanics, **191**
- Narrative, **189**
- Necessary, 155, 184
- Optional, 155, 187
- Representation, 158
- Rules, **191**
- Visuals, **186**
- Content-intensive, 19
- Convolutional neural network, 195, 249, 283
- Cost function, 32, 62
- Creative facets, 177, 184, 282
- Creative path, 178
- Creativity, 152, 191
- Crowdsourcing, 188, 200, 247, 272
- Culture, 218
- Darwinian evolution, 52
- Data analyst, 243
- Data corpus, 254
- Data mining, 213, 243, 288
- Data ownership, 289
- Data privacy, 288
- Data science, 18
- Data-driven, 25, 212, 230
- Deathmatch, 195
- Decision tree learning, 30, **68**
  - C4.5, 70
  - Gain ratio, 70
  - ID3, 69, 243
  - Information gain, 69
- Decision trees, 31, **68**, 231, 243
- Deep convolutional neural network, 120, 249, 251
- Deep fusion, 251
- Deep learning, 65, 86
- Deep Q network, 86, 106, 116, 145, 236
- Deep Q-learning, 88, 118, 129
- Deep reinforcement learning, 115, 131, 134
- Demographics, 218
- Dendrogram, 79, 240
- Depth, 192
- Designer, 156, 159, 174, 175, 178, 179, 187, 188, 190, 191, 198, 221, 263, 271
- Designer modeling, 275
- Deterministic, 45, 112
- Diamond-square algorithm, 170
- Doom, 106, 139, 141
- Drama management, 273
- Dynamic programming, 74, 76, 115
- Dynamic scripting, 118
- Economics, 31, 37
- Electrocardiography, 215

- Electrodermal activity, 215  
Electroencephalography, 215  
Electromyography, 215  
Eliza, 144  
Embodied conversational agents, 143, 144  
Emotion elicitation, 284  
Emotion recognition, 284  
Emotions, 37, 203  
End user, 260, 262  
Episodic memory, 226, 233  
Error function, 32, 62  
Evaluation function, 159  
Evolutionary algorithms, 49, 52, 97, 179  
Evolutionary computation, 30, **49**, 266  
    Covariance matrix adaptation evolution strategy, 55  
    Crossover, 52  
    Evolutionary strategies, 31, 54  
    Feasible-infeasible 2-population, 55, 198  
    Fitness function, 32, 49, 159  
    Generation, 53  
    Genetic algorithms, 31, 55  
    Genetic programming, 31  
    Inversion, 51  
    Multiobjective, 56  
    Mutation, 50  
    Natural evolution strategy, 55  
    Representation, 49  
Evolutionary planning, 113  
Evolutionary reinforcement learning, 117  
Evolutionary robotics, 117  
Experience replay, 116, 130  
Expert-knowledge systems, 32  
Exploration vs. exploitation, 104  
Expressive NPC, 143  
Expressive range, 158, 199  
Expressivity, 199  
Eye gaze, 129
- Facial expression, 214  
*Façade*, 190, 273, 287  
Feature selection, 247  
Fighting game AI competition, 146  
Fighting games, 146  
Finite state machines, 31, **33**  
    hierarchical, 33  
First-person shooters, 92, **139**, 196  
Flow channel, 209  
Fog of war, 102  
Forward model, 106, 281  
Fractals, 156  
Frequent pattern mining, 30, 77, **80**, 82  
    Apriori, 30, **80**  
    Generalized sequential patterns, 30, 81
- FSM, *see* Finite state machines  
Function approximator, 118  
Fuzzy logic, 134  
Fuzzy set, 37
- Galvanic skin response, 215  
Game adaptation, 275, 289  
Game affective loop, 284  
Game AI, 4  
    Ethical considerations, 287  
    Academia, 10  
    Areas, 259  
    Dominant methods, 261  
    Ethical considerations, 279  
    Frontier research, 26, 279, 291  
    General, 279  
    Industry, 11  
    Panorama, 26, 259  
    Secondary methods, 261  
Game analytics, 20, 25, 207  
Game context, 264  
Game controller, 18  
Game curator, 286  
Game data mining, 20, 25, 252, 260  
Game design, 25, 157, 195, 207, 211, 236, 242, 244, 271, 280, 287  
Game directing, 286  
Game experience questionnaire, 221  
Game facets, 282  
Game generation, **193**, 283  
Game industry, 37  
Game metrics, 206, 273  
Game stakeholder, 275  
Game studies, 189, 208  
Game testing, 207  
Game theory, 27, 31, 72  
Game user study, 200  
Games with a purpose, 141  
Gaussian process, 232  
Gaze tracking, 129, 216  
Gender, 218  
General emotional intelligence, 284  
General game affective loop, 284  
General game AI, 279  
General game design, 280  
General game generation, 283  
General game playing, 40, 108, 129, 280, 281  
General game playing competition, 23, 109, 281  
General general game AI, 280  
General intelligence, 23, 282, 284, 290  
General level design, 282  
General video game AI competition, 23, 113, **130**, 281

- General video game description language, 291  
 Generative adversarial networks, 171  
 Generative space, 199  
 Genetic programming, 117  
 Geneva wheel model, 226  
 Gesture, 18, 216  
 Global optimization, 52  
 Go, 16, 45, 93, 107, 112, 119  
     AlphaGo, 9, 16, 112  
 Grammars, 31, 163, 192  
     Grammatical evolution, 31  
 Graphics, 186  
 Graphs, 31  
 Grid-based path-planning, 41  
 Ground truth, 220, 234, 245, 285  
*GVGAI*, *see* General video game AI competition  
  
 Haptics, 18, 216  
*HCI*, *see* Human-computer interaction  
 Head pose, 215  
 Heart rate variability, 18, 250  
*Hearthstone*, 122  
 Heatmap, 199, 214  
 Hebbian learning, 59, 77  
 Heightmap, 169  
 Hidden information, 102  
 Hill climber, 50  
     Gradient-based, 50  
     Randomized, 50  
 Horror games, 217  
 Human computation, 142  
 Human-computer interaction, 203, 237, 264  
     Rich, 17, 204  
 Human-like, 95, 128, 143, 207, 236, 268, 281  
 Hybrid algorithms, 30, 82, 118, 208  
  
 Ideation tools, 286  
 Imitation, 207  
 Indie game development, 187, 270  
 Intelligent tutoring systems, 143  
 Intensity map, 169  
 Interaction design, 270  
 Interactive evolution, 178, 187  
 Interactive fiction, 144, 190, 216  
 Interactive narrative, 144, 271  
 Interactive storytelling, 189  
 Iterative refinement, 187  
 Iterative width search, 40  
  
 Killer application, 11, 22  
 Knowledge representation, 20  
  
 L-systems, 164  
  
 Learnability, 192, 236  
 Learning classifier systems, 118  
 Learning objective, 141  
 Level design, 177, 184, 282  
 Likert, 226  
 Linear discriminant analysis, 233  
 Linked data, 219  
 Local optimum, 51  
 Local search, 50  
 Logic programming, 161  
 Loss function, 32  
  
 Machine learning, 6, 11, 20, 32, 59, 71, 77,  
     108, 134, 148, 171, 206, 212, 230, 234,  
     273, 286  
 Macro-actions, 220  
 Mario AI competition, 41, 96, 110, 126, 266  
 Marketing, 207, 228  
 Markov decision process, 32, 72  
 Markov models, 173  
 Markov property, 72, 76, 100  
 MCTS, *see* Monte Carlo tree search  
 Memory-experience gap, 221  
 Micro-actions, 220  
 Midpoint displacement algorithm, 169  
*Minecraft*, 92, 141, 148, 169, 185, 288  
     Project Malmo, 148  
 Minimax, 8, 30, 42, 103, 109  
      $\alpha$ - $\beta$  pruning, 100  
 Mixed-initiative co-creation, 273  
 Mixed-initiative co-creativity, 201  
 Modalities of input, 215, 218  
 Monetization, 207, 243  
 Monte Carlo tree search, 30, 45, 112, 130, 174,  
     276, 281  
     Backpropagation, 45, 46  
     Expansion, 46  
     Rollout, 45, 112  
     Selection, 46  
     Simulation, 46  
     UCB1, 46  
 Moore neighborhood, 127  
 Motion tracking, 215  
*Ms Pac-Man*, 29, 33, 35, 38, 40, 42, 44, 47, 51,  
     56, 65, 68, 70, 76, 80, 82, 84, 87, 125  
     Competition, 149  
 Multi-player, 99  
 Multiagent systems, 27  
 Multimodal, 18  
 Multimodal fusion, 251  
 Muscle activation, 216  
  
 Natural language processing, 21, 144, 209  
 Navigation, 21, 41

- Navigation mesh, 41  
NEAT, *see* Neuroevolution of augmenting topologies  
Neuroevolution, 30, **83**, 84, 117, 129, 131, 140, 245, 287  
Neuroevolution of augmenting topologies, 84, 129, 133, 287  
Neuroscience, 208, 209  
Nintendo NES, 124  
Nintendo Wii, 95  
NLP, *see* Natural language processing  
No free lunch, 31, 58  
Noise, 169  
    Perlin noise, 171  
Non-deterministic, 44, 100  
Non-player character, 6, 14, 24, 99  
    Behavior, 110  
NP-hard, 16  
NPC, *see* Non-player character
- Objective function, 32  
Observability, 102  
One-and-a-half-player, 99  
Ontology, 123  
Open-source, 290  
OpenAI Universe, 132  
Opponent modeling, 205  
Optimization, 50, 68, 113, 189  
Orchestration, 195
- Pac-Man*, 29, **125**, 213  
Pairwise preference, 228, 234  
Path-planning, 110  
Pathfinding, 26, 41  
PCG, *see* Procedural content generation  
Peak-end rule, 226  
Perfect information, 102  
Personality, 218, 288  
Personality traits, 288  
Personalization, 236  
Personas, 96  
Phenomenological debugging, 242  
Photoplethysmography, 215  
Photorealism, 186  
Physical games, 211  
Physiological signal, 216, 254  
Physiology, 18, 209, 214, 274  
Planning, 21, 41, 113, 133, 146, 189, 262, 266  
    Hierarchical Task Network, 189  
Platformer experience dataset, 249, 253  
Platformers, 104, 126, 201  
Play for the experience, 92, 268  
Play to win, 92, 266  
Play traces, 245
- Player archetypes, 211  
Player behavior, 206  
Player behavior modeling, **220**, 260, 273  
Player character, 203  
Player experience, 25, 176, 180, 206, 212, 245, 273  
Player experience modeling, 180, 182, **220**, 234, 245, 260, 273  
Player expertise, 218  
Player metrics, 206, 214  
Player model  
    Input, 204, **213**  
    Methods, 204  
    Output, **219**  
Player model input  
    Context, **217**  
    Gameplay, **214**  
    Objective, **214**  
Player modeling, 9, 182, 203, 264, 268, 273  
    Ethics, 288  
    Hybrid algorithms, 213  
    Methods, **230**  
    Model-based, **208**, 208  
    Model-free, 208, **212**  
    Offline, 236  
    Runtime, 236  
Player profile, **218**  
Player profiling, 205  
Playtesting, 94, 133, 146, 200, 236, 268, 285  
Playthroughs, 20, 192, 267  
Poker, 45, 102, 121, 205  
Posture, 214, 216  
Potential fields, 134  
Practical wisdom, 31, 199  
Prediction, 207  
Preference, 232  
Preference learning, 58, **232**, 234, 247, 250  
    Neuroevolution, 182, 247  
    RankSVM, 71  
Preference learning toolbox, 233  
Preschooler games, 211  
Prey-predator games, 211, 249  
Probabilistic models, 31  
Procedural architecture, 184  
Procedural audio, 187  
Procedural content generation, 9, 151, 273  
    Assisted, 271  
    Autonomous, 156, 175, **180**, 270  
    Constructive, 156  
    Controllable, 156  
    Deterministic, 155  
    Evaluation, **197**  
    Experience-agnostic, 157, 175, **182**

- Experience-driven, 157, 175, **180**, 247, 275, 286  
Fractal, 169  
Generate-and-test, 156  
Grammar-based, 162  
Machine learning, 171, 195  
Mixed-initiative, 156, 162, 175, **176**, 271  
Noise, 169  
Non-controllable, 156  
Offline, 175  
Orchestration, 283  
Runtime, 175  
Search-based, 157  
Solver-based, 161  
Stochastic, 155  
Procedural personas, 27, 200, 236, 244, 275, 289  
Protocol design, 200  
Psychology, 203, 208, 228  
Adaptation level theory, 229  
Psychometrics, 234, 236  
Psychophysiology, 208, 249  
Pupil dilation, 215  
Pupillometry, 216  
Q-learning, 31, 73, **74**, 76, 116  
Discount factor, 75  
Learning rate, 75  
Racing games, 106, **136**  
TORCS, 106, 138  
Ranks, 226  
Ratings, 226, 233  
Inter-personal biases, 234  
Inter-personal differences, 227  
RBF networks, 60  
Real-time games, **104**  
Real-time strategy, 105  
Regression, 58, 231, 233  
Reinforcement learning, 8, 30, **71**, 85, 101, 108, 115, 230, 235, 266  
Backup, 74  
Bootstrapping, 74  
Exploration vs. exploitation, 73  
Policy, 72  
Reward, 32  
World model, 73  
ReLU, 88  
Representation, 30, 106  
Respiration, 216  
Retro Learning Environment, 131  
Reward, 71, 230, 236  
RL, *see* Reinforcement learning  
Rogue, 9, 151, 185  
Role-playing game, 93, 104  
RPG, *see* Role-playing game  
Rubber band AI, 95  
Sandbox games, 146  
Search tree, 39, 43  
Self-assessment manikin, 226  
Self-organizing map, **239**  
Self-reporting, 231  
Semantic memory, 226  
Semantics, 195  
*Sentient sketchbook*, 40, 177, 267  
Sentiment analysis, 209  
Serious games, **141**  
Simulated annealing, 51  
Simulated car racing championship, 138, 266  
Simulation-based testing, 94, 95, 267  
Single-player, 99  
Skin conductance, 250  
Sound  
Diegetic, 188  
Non-diegetic, 188  
SparCraft, 134  
Speech, 18, 214, 216  
StarCraft, 16, 21, 94, 102, **133**, 276  
TorchCraft, 17  
Steam, 253  
Stochasticity, **100**, 197  
Story generation, 189, 273  
Strategy games, **132**  
STRIPS, 115  
Super Mario Bros, 16, 47, 96, **126**, 246  
Supervised learning, 30, **57**, 108, 118, 230  
Generalization, 57  
Support vector machines, 30, **66**, 231  
Hard margin, 66  
Kernel, 67  
Maximum margin, 66  
RankSVM, 71  
Soft margin, 66  
Support vectors, 66  
Surprise search, 187  
Suspension of disbelief, 22  
Symbolic representation, 114  
Taxonomy, 73, 154, 204, 207, 263  
TD learning, 30, 31, 74, 87, 236, 266  
TD-Gammon, 8, 85, 116  
Telemetry, 20  
Text-based adventure games, 144, 216  
Theory of flow, 209  
Theory of fun, 211  
Think-aloud protocol, 221  
Tic-Tac-Toe, 8, 43

- Training signal, 32, 77, 232, 237  
Tree search, 20, **39**, 101, 103, 108, 109, 120  
    Classic tree search, 109  
Turing test, 96, 129, 140, 267, 268, 274, 281  
Turn-based games, **104**  
Turtle graphics, 164  
Two-player, 99  
  
U-matrix, 242  
Uninformed search, **40**  
    Breadth-first, 40  
    Depth-first, 40  
Unity, 39, 201  
Unreal engine, 39, 189  
Unsupervised learning, 30, **77**, 229, 237  
    Association mining, 207, 229  
    Clustering, **77**, 207, 229, 237  
    Frequent pattern mining, **80**, 238, 248  
    Hierarchical clustering, 30, 240  
    k-means, 30, 240  
    Self-organizing maps, 30, 59, 78, **239**  
Usability, 209  
User experience, 208, 273, 285  
User modeling, 203, 228  
  
User retention, 232, 243  
User studies, 200  
User testing, 237  
Utility, 30, 77, 158, 200, 245  
Utility-based AI, 32, **37**  
  
Video game description language, 130, 192  
Virtual agents, 143, 190  
Virtual camera, 189, 254  
Virtual cinematography, 144  
Virtual humans, 143  
Virtual reality, 216  
Visual cues, 248  
Visualization, 198  
VizDoom, 139, **141**  
Voxel, 169, 184  
  
Weapon generation, 160, 186, 195  
Weapon selection, 37  
Weka, 243  
WordNet, 195  
  
Z-machine, 145  
ZX Spectrum, 124

# Player Modeling

Georgios N. Yannakakis<sup>1</sup>, Pieter Spronck<sup>2</sup>, Daniele Loiacono<sup>3</sup>, and Elisabeth Andre<sup>4</sup>

- 1 Institute of Digital Games, University of Malta  
Msida, Malta  
[georgios.yannakakis@um.edu.mt](mailto:georgios.yannakakis@um.edu.mt)
- 2 Tilburg center of Cognition and Communication, University of Tilburg  
The Netherlands  
[p.spronck@uvt.nl](mailto:p.spronck@uvt.nl)
- 3 Politecnico di Milano  
Italy  
[daniele.loiacono@polimi.it](mailto:daniele.loiacono@polimi.it)
- 4 University of Augsburg  
Germany  
[andre@informatik.uni-augsburg.de](mailto:andre@informatik.uni-augsburg.de)

---

## Abstract

---

Player modeling is the study of computational models of players in games. This includes the detection, modeling, prediction and expression of human player characteristics which are manifested through cognitive, affective and behavioral patterns. This chapter introduces a holistic view of player modeling and provides a high level taxonomy and discussion of the key components of a player's model. The discussion focuses on a taxonomy of approaches for constructing a player model, the available types of data for the model's input and a proposed classification for the model's output. The chapter provides also a brief overview of some promising applications and a discussion of the key challenges player modeling is currently facing which are linked to the input, the output and the computational model.

**1998 ACM Subject Classification** I.2.1 Applications and Expert Systems: Games

**Keywords and phrases** User modeling, Computer Games, Computational and Artificial Intelligence, Affective Computing,

**Digital Object Identifier** 10.4230/DFU.xxx.yyy.p

## 1 Introduction

Digital games are dynamic, *ergodic* media (i.e., a user interacts with and alters the state of the medium). They are designed to be highly engaging and embed rich forms of user interactivity. Collectively, the human-computer interaction (HCI) attributes of digital games allow for high levels of player incorporation [9]. As such, they yield dynamic and complex emotion manifestations which cannot be captured trivially by standard methods in affective computing or cognitive modeling research. The high potential that games have in affecting players is mainly due to their ability of placing the player in a continuous mode of interaction, which develops complex cognitive, affective and behavioral responses. The study of the player in games may not only contribute to the design of improved forms of HCI, but also advance our knowledge of human experiences.

Every game features at least one user (i.e., the player), who controls an avatar or a group of miniature entities in a virtual/simulated environment [9]. Control may vary from the

relatively simple (e.g., limited to movement in an orthogonal grid) to the highly complex (e.g., having to decide several times per second between hundreds of different possibilities in a highly complex 3D world). The interaction between the player and the game context is of prime importance to modern game development, as it breeds unique stimuli which yield emotional manifestations to the player. Successful games manage to elicit these emotions in a manner that is appreciated by the player, and which form the main reason that the player is willing to engage in the game [50].

The primary goal of player modeling and player experience research is to understand how the interaction with a game is experienced by individual players. Thus, while games can be utilized as an arena for eliciting, evaluating, expressing and even synthesizing experience, we argue that one of the main aims of the study of players in games is the understanding of players' cognitive, affective and behavioral patterns. Indeed, by the nature of what constitutes a game, one cannot dissociate games from player experience.

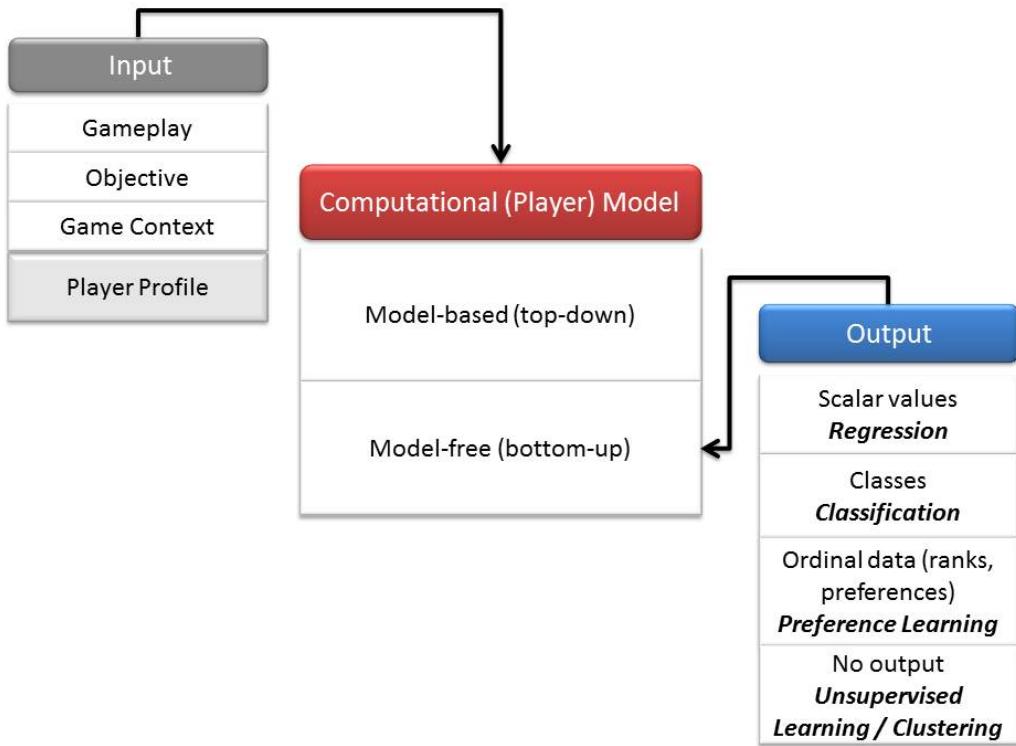
This chapter focuses on experience aspects that can be detected from, modeled from, and expressed in games with human players. We explicitly exclude the modeling of non-player characters (NPCs), as in our view, player modeling involves a *human player* in the modeling process. We also make a distinction between *player modeling* [10, 26] and *player profiling*. The former refers to modeling complex *dynamic* phenomena during gameplay interaction, whereas the latter refers to the categorization of players based on *static* information that does not alter during gameplay — that includes personality, cultural background, gender and age. We will mainly focus on the first, but will not ignore the second, as the availability of a good player profile may contribute to the construction of reliable player models.

In summary, player modeling — as we define it in this chapter — is the study of computational means for the modeling of player cognitive, behavioral, and affective states which are based on data (or theories) derived from the interaction of a human player with a game [78]. Player models are built on *dynamic* information obtained during game-player interaction, but they could also rely on *static* player profiling information. Unlike earlier studies focusing on taxonomies of *behavioral* player modeling — e.g., via a number of different dimensions [58] or direct/indirect measurements [56] — we view player modeling in a holistic manner including cognitive, affective, personality and demographic aspects of the player. Moreover, we exclude approaches that are not directly based on human-generated data or not based on empirically-evaluated theories of player experience, human cognition, affect or behavior. The chapter does not intend to provide an inclusive review of player modeling studies under the above definition, but rather a high-level taxonomy that explores the possibilities with respect to the modeling approach, the model's input and the model's output.

The rest of this chapter provides a taxonomy and discussion of the core components of a player model which are depicted in Figure 1. That includes the computational model itself and methods to derive it as well as the model's input and output. The chapter illustrates a few promising applications of player modeling and ends with a discussion on open research questions for future research in this field.

## **2 Computational model**

Player modeling is, primarily, the study and use of artificial and computational intelligence (AI and CI) techniques for the construction of computational models of player behavior, cognition and emotion, as well as other aspects beyond their interaction with a game (such as their personality and cultural background). Player modeling places an AI umbrella to



■ **Figure 1** Player Modeling: the core components.

the multidisciplinary intersection of the fields of user (player) modeling, affective computing, experimental psychology and human-computer interaction.

One can detect behavioral, emotional or cognitive aspects of either a human player or a non-player character (NPC). In principle, there is no need to model an NPC, for two reasons: (1) an NPC is coded, therefore a perfect model for it exists in the game's code, and is known by the game's developers; and (2) one can hardly say that an NPC possesses actual emotions or cognition. However, NPC modeling can be a useful testbed for player modeling techniques, by comparing the model discovered with the actual coded one. More interestingly, it can be an integral component of adaptive AI that changes its behavior in response to the dynamics of the opponents [6]. Nevertheless, while the challenges faced in modeling NPCs are substantial, the issues raised from the modeling of human players define a far more complex and important problem for the understanding of player experience.

By clustering the available approaches for player modeling, we are faced with either *model-based* or *model-free* approaches [80] as well as potential *hybrids* between them. The remaining of this section presents the key elements of both model-based and model-free approaches.

## 2.1 Model-based (top-down) approaches

According to a *model-based* or top-down [80] approach a player model is built on a theoretical framework. As such, researchers follow the modus operandi of the humanities and social sciences, which hypothesize models to explain phenomena, usually followed by an empirical phase in which they experimentally determine to what extent the hypothesized models fit

observations.

Top-down approaches to player modeling may refer to emotional models derived from **emotion theories** (e.g., cognitive appraisal theory [21]). Three examples are: (1) the emotional dimensions of arousal and valence [19], (2) Frome's comprehensive model of emotional response to a single-player game [22], and (3) Russell's circumplex model of affect [51], in which emotional manifestations are mapped directly to specific emotional states (e.g., an increased heart rate of a player may correspond to high arousal and therefore to excitement). Model-based approaches can also be inspired by a general theoretical framework of behavioral analysis and/or cognitive modeling such as usability theory [30], the belief-desire-intention (BDI) model, the cognitive theory by Ortony, Clore, & Collins [47], Skinner's model [57], or Scherer's theory [53]. Moreover, theories about user affect exist that are driven by game design, such as Malone's design components for 'fun' games [40] and Koster's theory of 'fun' [35], as well as game-specific interpretations of Csikszentmihalyi's concept of Flow [14]. Finally, several top-down difficulty and challenge measures [2, 28, 45, 59, 60, 70, 75] have been proposed for different game genres as components of a player model. In all of these studies, difficulty adjustment is performed based on a player model that implies a direct link between challenge and 'fun'.

Note, however, that even though the literature is rich in theories on emotion, caution is warranted with the application of such theories to games (and game players), as most of them have not been derived from or tested on ergodic media such as games. Calleja [9], for instance, reflects on the inappropriateness of the concepts of 'flow', 'fun' and 'magic circle' (among others) for games.

## **2.2 Model-free (bottom-up) approaches**

Model-free approaches refer to the construction of **an unknown mapping (model)** between (player) input and a player state representation. As such, model-free approaches follow the modus operandi of the exact sciences, in which observations are collected and analyzed to generate models without a strong initial assumption on what the model looks like or even what it captures. Player data and annotated player states are collected and used to derive the model. Classification, regression and preference learning techniques adopted from machine learning or statistical approaches are commonly used for the construction of a computational model. Data clustering is applied in cases where player states are not available.

In model-free approaches we meet attempts to model and predict player actions and intentions [64, 65, 76] as well as *game data mining* efforts to identify different behavioral playing patterns within a game [15, 43, 61, 62, 72]. Model-free approaches are common for facial expression and head pose recognition since subjects are asked to annotate facial (or head pose) images of users with particular affective states (see [55] among others) in a crowd-sourcing fashion. The approach is also common in studies of psychophysiology in games (see [68, 79] among others).

## **2.3 Hybrid approaches**

The space between a completely model-based and a completely model-free approach can be viewed as a continuum along which any player modeling approach might be placed. While a completely model-based approach relies solely on a theoretical framework that maps a player's responses to game stimuli, a completely model-free approach assumes there is an unknown function between modalities of user input and player states that a machine learner (or a statistical model) may discover, but does not assume anything about the structure

of this function. Relative to these extremes, the vast majority of the existing works on player modeling may be viewed as hybrids between the two ends of the spectrum, containing elements of both approaches.

### 3 Input

The model's input can be of three main types: (1) anything that a human player is doing in a game environment gathered from *gameplay* data (i.e., behavioral data); (2) *objective* data collected as bodily responses to game stimuli such as physiology and body movements; and (3) the *game context* which comprises of any player-agent interactions but also any type of game content viewed, played, and/or created. The three input types are detailed in the remainder of this section. We also discuss static information on the player (such as personality and gender) that could feed a player model.

#### 3.1 Gameplay input

The main assumption behind the use of behavioral (gameplay-based) player input is that player actions and real-time preferences are linked to player experience as games may affect the player's cognitive processing patterns and cognitive focus. In the same vein, cognitive processes may influence player experience. One may infer the player's present experience by analyzing patterns of his interaction with the game, and by associating his emotions with context variables [12, 24]. Any element derived from the interaction between the player and the game forms the basis for gameplay-based player modeling. This includes detailed attributes of the player's behavior (i.e., game metrics) derived from responses to system elements (i.e., NPCs, game levels, or embodied conversational agents). Game metrics are statistical spatio-temporal features of game interaction [17]. Such data is usually mapped to levels of cognitive states such as attention, challenge and engagement [12]. In addition, both general measures (such as performance and time spent on a task) and game-specific measures (such as the weapons selected in a shooter game — e.g., in [25]) are relevant.

A major problem with interpreting such behavioral player input is that the actual player experience is only indirectly observed by measuring game metrics. For instance, a player who has little interaction with a game might be thoughtful and captivated, or just bored and busy doing something else. Gameplay metrics can only be used to approach the likelihood of the presence of certain player experiences. Such statistics may hold for player populations, but may provide little information for individual players. Therefore, when one attempts to use pure *gameplay* metrics to make estimates on player experiences and make the game respond in an appropriate manner to these perceived experiences, it is advisable to keep track of the feedback of the player to the game responses, and adapt when the feedback indicates that the player experience was gauged incorrectly.

#### 3.2 Objective input

Games can elicit player emotional responses which, in turn, may affect changes in the player's physiology, reflect on the player's facial expression, posture and speech, and alter the player's attention and focus level. Monitoring such bodily alterations may assist in recognizing and synthesizing the player's model. As such, the objective approach to player modeling incorporates access to multiple modalities of player input.

Within objective player modeling, a number of real-time recordings of the player may be investigated. Several studies have explored the interplay between physiology and gameplay by

investigating the impact of different gameplay stimuli to dissimilar physiological signals. Such signals are usually obtained through electrocardiography (ECG) [79], photoplethysmography [68, 79], galvanic skin response (GSR) [41, 49], respiration [68], electroencephalography (EEG) [44], electromyography (EMG) and pupillometry [7]. In addition to physiology one may track the player’s bodily expressions (motion tracking) at different levels of detail and infer the real-time affective responses from the gameplay stimuli. The core assumption of such input modalities is that particular bodily expressions are linked to basic emotions and cognitive processes. Motion tracking may include body posture [52] and head pose [55], as well as gaze [4] and facial expression [48].

While such objective multimodal measurements are usually more meaningful than the gameplay inputs discussed in the previous subsection, a major problem with most of them is that they are viewed by the player as invasive, thus affecting the player’s experience with the game. This effect should be taken into account when interpreting measurements.

### **3.3 Game context input**

In addition to both gameplay and objective input, the game’s context is a necessary input for player modeling. Game context refers to the real-time parameterized state of the game. Player states are always linked to game context; a player model that does not take context into account runs a high risk of inferring erroneous states for the player. For example, an increase in galvanic skin response (GSR) can be linked to a set of dissimilar high-arousal affective states such as frustration and excitement; thus, the cause of the GSR increase (e.g., a player’s death or level completion) needs to be fused within the GSR signal and embedded in the model.

### **3.4 Player profile information**

Differences between players lead to different playing styles and preferences. Player profile information includes all the information about the player which is static and it is not directly (nor necessarily) linked to gameplay. This may include information on player personality (such as expressed by the Five Factor Model of personality [13]), culture dependent variables, and general demographics such as gender and age. Such information may be used as static model input and could lead to the construction of more accurate player models.

A typical way of employing player profile information is the stereotype approach [34]. The approach attempts to (automatically) assign a player to a previously defined subgroup of the population, for which key characteristics have been defined. Each subgroup is represented by a stereotype. After identifying to which subgroup the player belongs, the game can choose responses appropriate for the corresponding stereotype. This approach has been used by Yannakakis & Hallam [73] and by Thue [63]. They define stereotypes in terms of a player’s gaming profile, rather than the gamer’s characteristics outside the realm of gaming.

Van Lankveld et al [69, 71] look beyond pure gaming behavior, attempting to express a player’s personality profile in terms of the Five Factor Model. While they achieve some success in modeling players in terms of the five factors, they notice that gameplay behavior not necessarily corresponds to “real life” behavior, e.g., an introverted person may very well exhibit in-game behavior that would typically be assigned to an extroverted person. Therefore, we can recognize two caveats which should be taken into account when aiming to use a real-life personality profile of a player to construct a model of the player inside a game: (1) the player’s behavior inside the game not necessarily corresponds to his real-life

behavior and vice versa; and (2) a player’s real-life profile not necessarily indicates what he appreciates in a game.

## 4 Output

The model’s output is usually a set of particular *player states*. Such states can be represented as a class, a scalar (or a vector of numbers) that maps to a player state — such as the emotional dimensions of arousal and valence or a behavioral pattern — or a relative strength (preference). The output of the model is provided through an annotation process which can either be driven by self-reports or by reports expressed indirectly by experts or external observers [80]. However, there are instances where reports on player states are not available; output then must be generated by unsupervised learning (see [15, 17] among others).

The most direct way to annotate a player state is to ask the players themselves about their playing experience and build a model based on these annotations. Subjective player state annotations can be based on either a player’s free-response during play or on forced data retrieved through *questionnaires*. Free-response naturally contains richer information about the player’s state, but it is often unstructured, even chaotic, and thus hard to analyze appropriately. Forcing players to self-report their experiences using directed questions, on the other hand, constrains them to specific questionnaire items which could vary from simple tick boxes to multiple choice items. Both the questions and the answers provided may vary from single words to sentences. Questionnaires can either involve elements of the player experience (e.g., the *Game Experience Questionnaire* [29]), or demographic data and/or personality traits (e.g., a validated psychological profiling questionnaire such as the NEO-PI-R [13]).

Alternatively, experts or external observers may annotate the player’s experiences in a similar fashion. Third-person annotation entails the identification of particular player states (given in various types of representation as we will see below) by player experiences and game design experts (or crowd-sourced via non-experts). The annotation is usually based on the triangulation of multiple modalities of player and game input, such as the player’s head pose, in-game behavior and game context [55]. Broadly-accepted emotional maps such as the Facial Action Coding System [18] provide common guidelines for third-person emotion annotation.

Three types of annotations (either forced self-reports or third-person reports) can be distinguished. The first is the *rating-based* format [41], which labels player experience states with a scalar value or a vector of values (found, for instance, in the Game Experience Questionnaire [29]). The second is the *class-based* format, which asks subjects to pick a user state from a particular representation which could vary from a simple boolean question (*was that game level frustrating or not? is this a sad facial expression?*) to a user state selection from, for instance, the Geneva Emotion Wheel [54]. The third is the *preference-based* format, which asks subjects to compare an experience in two or more variants/sessions of the game [77] (*was that level more engaging than this level? which facial expression looks happier?*). A recent comparative study has exposed the limitations of rating approaches over ranking questionnaire schemes (e.g., pairwise preference) which include increased order of play and inconsistency effects [74].

Beyond annotated player states or player profiles (such as personality traits), player models may be constructed to predict attributes of gameplay (e.g., in [39, 65] among others) or objective manifestations of the experience [3].

## 5 Applications

In this section we identify and briefly illustrate a few promising, and certainly non inclusive, applications of player modeling to game design and development, that range from adapting the challenge during the game to personalizing the purchasing model in free-to-play games.

### 5.1 Adaptive Player Experience and Game Balancing

As already mentioned in section 2.1 there exist several theoretical frameworks investigating the relationship between experience and game interaction that have been either built with primarily games in mind (e.g. the player immersion model of Calleja [9]; the theory of ‘fun’ of Koster [35]) or derived from other fields (e.g. psychology) and domains and tailored to games (e.g. the theory of flow [14] adopted for games [11]). Essentially what unifies all these theories is that ultimate player experience is achieved when elements of the game (mechanics, storyline, challenges) are in some sort of right balance with the general skills of the player. A common, but rather simplistic, approach to balance between game challenges and different player skills in order to match a broader range of players consists of providing a small, predefined set of difficulty levels which the player can pick from. A more sophisticated approach consists of adapting the game’s challenge in response to the actions of the players and to the state of the game environment; relevant examples of this approach are the *AI director* [8] of *Left 4 Dead* (Valve, 2008) and the *race script* [23] framework used in *Pure* (Black Rock Studio, 2008).

Most of the game balancing approaches currently used rely on simple in-game statistics to estimate the state of the players and make strong assumptions on what kind of experience the players are looking for in the game (e.g., the basic assumption behind the AI director in *Left 4 Dead* is that players enjoy dramatic and unpredictable changes of pace). While, in general, such assumptions hold for a large number of players (as the designers usually have a good idea of their players), they are not universally applicable and may actually exclude large groups of potential players that would be willing to play a game if it would offer an experience more to their liking. Reliable player modeling techniques have the potential to adapt the challenge level of a game (and other gameplay elements) in a manner that suits each individual player [75, 1].

### 5.2 Personalized Game Content Generation

Procedural content generation (PCG) aims to deliver a large amount of game content algorithmically with limited memory resources. Nowadays PCG is mainly used to cut the development cost and, at the same time, to face the increasing expectations of players in terms of game content. Recently, the problem of automating the generation of high-quality game content has attracted considerable interest in the research community (see [78] and the corresponding chapter in this volume). In particular, research showed that search algorithms can be combined successfully with procedural content generation to discover novel and enjoyable game content [38, 66, 67]. Accordingly, the automation of content creation offers an opportunity towards realizing player model-driven procedural content generation in games [80]. The coupling of player modeling with PCG approaches may lead to the automatic generation of personalized game content.

Player models may also inform the generation of computational narrative (viewed here as a type of game content [80]). Predictive models of playing behavior, cognition and affect can drive the generation of individualised scenarios in a game. Examples of the coupling between

player modeling and interactive narrative include the affect-driven narrative systems met in *Façade* [42] and *FearNot!* [5], the emotion-driven narrative building system in *Storybricks* (Namaste Entertainment, 2012), and the affect-centred game narratives such as the one of *Final Fantasy VII* (Square Product, 1997).

### 5.2.1 Towards Believable Agents

Human player models can inform and update believable agent architectures. Behavioral, affective and cognitive aspects of human gameplay can improve the human-likeness and believability of any agent controller — whether that is ad-hoc designed or built on data derived from gameplay. While the link between player modelling and believable agent design is obvious and direct the research efforts towards this integration within games is still sparse. However, the few efforts made on the imitation of human game playing for the construction of believable architectures have resulted in successful outcomes. Human behavior imitation in platform [46] and racing games [31] have provided human-like and believable agents while similar approaches for developing *Unreal Tournament* bots (e.g. in [33]) recently managed to pass the Turing test in the 2k BotPrize competition.

## 5.3 Playtesting Analysis and Game Authoring

While aiming at creating a particular game *experience*, designers can only define and alter the game *mechanics* [11, 27] (i.e., game rules) that, in turn, will affect the playing experience. From the mechanics arise the *game dynamics*, i.e., *how* the game is actually played. The dynamics lead to *game aesthetics*, i.e., what the player experiences during the game.

Even when a game is tested specifically to determine whether it provides the desired player experience, it is usually difficult to identify accurately which are the specific elements of the mechanics that work as intended. Player modeling, which yields a relationship between player state, game context, and in-game behavior, may support the analysis of playtesting sessions and the identification of what appeals to a particular player, and what does not (see [15, 43, 72] among many).

Player modeling provides a multifaceted improvement to game development as it does not only advance the study of human play and the enhancement of human experience. Quantitative testing via game metrics — varying from behavioral data mining to in-depth low scale studies — is improved as it complements existing practices [78].

Finally, user models can enhance authoring tools that, in turn, can assist the design process. The research field that bridges user modeling and AI-assisted design is in its infancy and only a few example studies can be identified. Indicatively, designer models have been employed to personalise mixed-initiative design processes [37, 36]. Such models drive the procedural generation of designer-tailored content.

## 5.4 Monetization of free-to-play games

Recent years have seen an increasing number of successful free-to-play (F2P) games on diverse platforms, including Facebook, mobile devices and PC. In F2P games, playing the game itself is free. Revenues come from selling additional contents or services to the players with in-game *microtransactions* [20]. In order to be profitable, these games require developers to constantly monitor the purchasing behavior of their players [20]. Player modeling may improve the understanding of the players behavior [16] and help with the identification of players who are willing to pay. In addition, the information provided by player modeling might be used to customize the market content and mechanisms, eventually leading to increased profits.

## 6 The road ahead: challenges and questions

In this section we list a number of critical current and future challenges for player modeling as well as promising research directions, some of which have been touched upon in the previous sections.

- Regardless of the line of research in player modeling chosen, the biggest obstacle right now is lack of proper and rich data publicly available to the researchers. What is required is a rich multimodal corpus of gameplay and player data as well as player descriptions. Such a corpus must include detailed gameplay data for several games for a large number of players, including actions, events, locations, timestamps as well as biometrical data, that are trivial to obtain in large volumes (e.g., camera images and eye-tracking). Demographic data for the players must be available, as well as player information in the form of several questionnaires and structured interview data. Not all this data needs to be available for every subject in the database; several large datasets of gameplay data already exist, and it would be beneficial to include those in the database too.
- The use of procedural content generation techniques for the design of better games has reached a peak of interest in commercial and independent game development [78], which is showcased by successful (almost entirely procedurally generated) games such as *Minecraft* (Mojang, 2011) and *Love* (Eskil Steenberg, 2010). Future games, in general, are expected to contain less manual and more user-generated or procedurally-generated content, as the cost of content creation and the content creation bottleneck are key challenges for commercial game production. As the number of games that are (partially or fully) automatically generated grows, the challenge of modeling players in never-ending open worlds of infinite replayability value increases substantially.
- Nowadays, several modalities or player input are still implausible within commercial game development. For instance, existing hardware for physiological measurements requires the placement of body parts (e.g., head or fingertips) to the sensors, making physiological signals such as EEG, respiration and skin conductance rather impractical and highly intrusive for most games. Modalities such as facial expression and speech could be technically plausible in future games: even though the majority of the vision-based affect-detection systems currently available cannot operate in real-time [81], the technology in this field is rapidly evolving [32].

On a positive note, recent advances in sensor technology have resulted in low-cost unobtrusive biofeedback devices appropriate for gaming applications. In addition, top game developers have started to experiment with multiple modalities of player input (e.g., physiological and behavioral patterns) for the personalization of experience of popular triple-A games such as *Left 4 Dead* (Valve, 2008) [1]. Finally, recent technology advances in gaming peripherals such as the PrimeSense camera showcase a promising future for multimodal natural interaction in games.

- Comparing model-based and model-free approaches to player modeling, we note that model-based inherently contains argumentation and understanding for the choices of the model, which model-free lacks. However, practice shows that model-based approaches often fail to encompass relevant features because of a lack of insight of the model builders. The model-free approach has the advantage of automatically detecting relevant features; however, it is also prone to detecting meaningless relationships between user attributes, game context and user experience. In computer games an extensive set of features of player behavior can be extracted and measured. At the same time there is, usually, lack of insight in what these features actually mean, at least at present. Therefore, in the

current state of research, model-free approaches seem most suitable. Domain-specific knowledge, feature extraction and feature selection are necessary to achieve meaningful models of players .

- As mentioned before, player characteristics within a game environment may very well differ from the characteristics of the player when dealing with reality. Thus, validated personality models such as psychology's Five Factor Model might not fit well to game behavior. An interesting direction in player modeling research is to determine a fundamental personality model for game behavior; such a model will have some correspondence with the Five Factor Model, but will also encompass different characteristics. Moreover, the behavioral clues that can be found in game behavior may be considerably different from those that can be found in reality.
- Experience has shown that diligent application of data mining techniques may provide insight into group behaviors. However, it remains difficult to make predictions about individuals. As such, player models can usually only give broad and fuzzy indications on how a game should adapt to cater to a specific player. One possible solution is to define several possible player models and classify an individual player as one of them (the stereotyping approach). Then, when gameplay is going on, the model can be changed in small steps to fit the player better. I.e., the player model is not determined as a static representation of the player, used to determine how the game should be adapted; rather it is a dynamic representation of a group of players, that changes to highlight the general characteristics of a specific player, and drives the game adaptation dynamically. In our view, this step-wise approach to game adaption by means of player modeling has the potential to lead to quick, good results in creating games that offer a personalized form of engagement to the player.

---

## References

---

- 1 M. Ambinder. Biofeedback in gameplay: How valve measures physiology to enhance gaming experience. In *Game Developers Conference*, 2011.
- 2 Gustavo Andrade, Geber Ramalho, Hugo Santana, and Vincent Corruble. Extending reinforcement learning to provide dynamic game balancing. In *Proceedings of the Workshop on Reasoning, Representation, and Learning in Computer Games, 19th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 7–12, August 2005.
- 3 S. Asteriadis, K. Karpouzis, N. Shaker, and G.N. Yannakakis. Does your profile say it all? using demographics to predict expressive head movement during gameplay.
- 4 Stylianos Asteriadis, Kostas Karpouzis, and Stefanos D. Kollias. A neuro-fuzzy approach to user attention recognition. In *Proceedings of ICANN*, pages 927–936. Springer, 2008.
- 5 Ruth Aylett, Sandy Louchart, Joao Dias, Ana Paiva, and Marco Vala. Fearnnot!—an experiment in emergent narrative. In *Intelligent Virtual Agents*, pages 305–316. Springer, 2005.
- 6 Sander Bakkes, Pieter Spronck, and Jaap van den Herik. Opponent modeling for case-based adaptive game ai. *Entertainment Computing*, 1(1):27–37, 2009.
- 7 A. Barreto, J. Zhai, and M. Adjouadi. Non-intrusive physiological monitoring for automated stress detection in human-computer interaction. In *Proceedings of Human Computer Interaction*, pages 29–39. Springer, 2007.
- 8 Michael Booth. The ai systems of left 4 dead. In *Keynote, Fifth Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE09)*, 2009.
- 9 G. Calleja. *In-Game: From Immersion to Incorporation*. The MIT Press, 2011.

- 10 D. Charles and M. Black. Dynamic player modelling: A framework for player-centric digital games. In *Proceedings of the International Conference on Computer Games: Artificial Intelligence, Design and Education*, pages 29–35, 2004.
- 11 Jenova Chen. Flow in games (and everything else). *Commun. ACM*, 50(4):31–34, April 2007.
- 12 C. Conati. Probabilistic Assessment of User’s Emotions in Educational Games. *Journal of Applied Artificial Intelligence, special issue on “Merging Cognition and Affect in HCI”*, 16:555–575, 2002.
- 13 Paul T. Costa and Robert R. McCrae. Domains and facets: hierarchical personality assessment using the revised NEO personality inventory. *Journal of Personality Assessment*, 64(1):21–50, 1995.
- 14 Mihaly Csikszentmihalyi. *Flow: the Psychology of Optimal Experience*. Harper Collins, 1990.
- 15 A. Drachen, A. Canossa, and G. N. Yannakakis. Player Modeling using Self-Organization in Tomb Raider: Underworld. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games*, pages 1–8, Milan, Italy, September 2009. IEEE.
- 16 A. Drachen, R. Sifa, C. Bauckhage, and C. Thurau. Guns, swords and data: Clustering of player behavior in computer games in the wild. In *Computational Intelligence and Games (CIG), 2012 IEEE Conference on*, pages 163–170, 2012.
- 17 A. Drachen, C. Thurau, J. Togelius, and G. N. Yannakakis. *Large-scale Data Mining in Games*. Springer-Verlag, 2012.
- 18 P. Ekman and W.V. Friesen. Facial action coding system: A technique for the measurement of facial movement. palo alto. *CA: Consulting Psychologists Press. Ellsworth, PC, & Smith, CA (1988). From appraisal to emotion: Differences among unpleasant feelings. Motivation and Emotion*, 12:271–302, 1978.
- 19 L. Feldman. Valence focus and arousal focus: Individual differences in the structure of affective experience. *Journal of Personality and Social Psychology*, 69:53–166, 1995.
- 20 Tim Fields and Brandon Cotton. *Social Game design: monetization methods and mechanics*. Morgan Kaufmann, 2011.
- 21 N. Frijda. *The Emotions*. Cambridge University Press, Engelwood cliffs, NJ, 1986.
- 22 J. Frome. Eight ways videogames generate emotion. In *Proceedings of DiGRA 2007*, 2007.
- 23 Iain Gilfeather. Advanced racing game ai in pure. *GDC Europe*, 2009.
- 24 J. Gratch and S. Marsella. Evaluating a computational model of emotion. *Autonomous Agents and Multi-Agent Systems*, 11(1):23–43, 2005.
- 25 Erin Hastings, Ratan Guha, and Kenneth O. Stanley. Evolving content in the galactic arms race video game. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games (CIG)*, 2009.
- 26 R. Houlette. *Player Modeling for Adaptive Games. AI Game Programming Wisdom II*, pages 557–566. Charles River Media, Inc, 2004.
- 27 Robin Hunicke, Marc LeBlanc, and Robert Zubek. Mda: A formal approach to game design and game research. In *Proceedings of the AAAI Workshop on Challenges in Game AI*, pages 04–04, 2004.
- 28 Hiroyuki Iida, N. Takeshita, and J. Yoshimura. A metric for entertainment of boardgames: its implication for evolution of chess variants. In R. Nakatsu and J. Hoshino, editors, *IWEC2002 Proceedings*, pages 65–72. Kluwer, 2003.
- 29 W. IJsselsteijn, K. Poels, and YAW de Kort. The game experience questionnaire: Development of a self-report measure to assess player experiences of digital games. *TU Eindhoven, Eindhoven, The Netherlands*, 2008.
- 30 K. Isbister and N. Schaffer. *Game Usability: Advancing the Player Experience*. Morgan Kaufman, 2008.

- 31 German Gutierrez Jorge Munoz and Araceli Sanchis. Towards imitation of human driving style in car racing games. In Philip Hingston, editor, *Believable Bots: Can Computers Play Like People?* Springer, 2012.
- 32 David Kadish, Nikolai Kummer, Aleksandra Dulic, and Homayoun Najjaran. The empathy machine. In Marc Herrlich, Rainer Malaka, and Maic Masuch, editors, *Entertainment Computing - ICEC 2012*, volume 7522 of *Lecture Notes in Computer Science*, pages 461–464. Springer Berlin Heidelberg, 2012.
- 33 Igor V Karpov, Jacob Schrum, and Risto Miikkulainen. Believable bot navigation via playback of human traces. In Philip Hingston, editor, *Believable Bots: Can Computers Play Like People?* Springer, 2012.
- 34 A. Kobsa. User modeling: Recent work, prospects and hazards. *Human Factors in Information Technology*, 10:111–125, 1993.
- 35 Raph Koster. *A theory of fun for game design*. Paraglyph press, 2005.
- 36 Antonios Liapis, Georgios Yannakakis, and Julian Togelius. Sentient sketchbook: Computer-aided game level authoring. In *Proceedings of ACM Conference on Foundations of Digital Games*, 2013. In Print.
- 37 Antonios Liapis, Georgios N. Yannakakis, and Julian Togelius. Adapting models of visual aesthetics for personalized content creation. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(3):213–228, 2012.
- 38 Daniele Loiacono, Luigi Cardamone, and Pier Luca Lanzi. Automatic track generation for high-end racing games using evolutionary computation. *Computational Intelligence and AI in Games, IEEE Transactions on*, 3(3):245–259, 2011.
- 39 T. Mahlmann, A. Drachen, J. Togelius, A. Canossa, and G.N. Yannakakis. Predicting player behavior in tomb raider: Underworld. In *Computational Intelligence and Games (CIG), 2010 IEEE Symposium on*, pages 178–185. IEEE, 2010.
- 40 Thomas W. Malone. What makes things fun to learn? heuristics for designing instructional computer games. In *Proceedings of the 3rd ACM SIGSMALL symposium and the first SIGPC symposium on Small systems*, pages 162–169, 1980.
- 41 R. L. Mandryk, K. M. Inkpen, and T. W. Calvert. Using Psychophysiological Techniques to Measure User Experience with Entertainment Technologies. *Behaviour and Information Technology (Special Issue on User Experience)*, 25(2):141–158, 2006.
- 42 Michael Mateas and Andrew Stern. Façade: An experiment in building a fully-realized interactive drama. In *Game Developers Conference, Game Design track*, volume 2, page 82, 2003.
- 43 Olana Missura and Thomas Gärtner. Player modeling for intelligent difficulty adjustment. In Johannes Fürnkranz Arno Knobbe, editor, *Proceedings of the ECML-09 Workshop From Local Patterns to Global Models (LeGo-09)*, Bled, Slovenia, September 2009.
- 44 Anton Nijholt. BCI for Games: A State of the Art Survey. In *Proceedings of Entertainment Computing - ICEC 2008*, pages 225–228, 2009.
- 45 Jacob Kaae Olesen, Georgios N. Yannakakis, and John Hallam. Real-time challenge balance in an RTS game using rtNEAT. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games*, pages 87–94, Perth, Australia, December 2008. IEEE.
- 46 Juan Ortega, Noor Shaker, Julian Togelius, and Georgios N Yannakakis. Imitating human playing styles in super mario bros. *Entertainment Computing*, 2012.
- 47 A. Ortony, G. L. Clore, and A. Collins. *The Cognitive Structure of Emotions*. Cambridge University Press, 1988.
- 48 Maja Pantic and George Caridakis. *Emotion-Oriented Systems: The Humaine Handbook*, chapter Image and Video Processing for Affective Applications, pages 101–117. Springer-Verlag Berlin Heidelberg, 2011.

- 49 P. Rani, N. Sarkar, and C. Liu. Maintaining optimal challenge in computer games through real-time physiological feedback. In *Proceedings of the 11<sup>th</sup> International Conference on Human Computer Interaction*, 2005.
- 50 N. Ravaja, M. Salminen, J. Holopainen, T. Saari, and J.J.A. Laarni. Emotional response patterns and sense of presence during video games: potential criterion variables for game design. In *Proceedings of the third Nordic Conference on Human-Computer Interaction*, 2004.
- 51 J. A. Russell. Core affect and the psychological construction of emotion. *Psychological Rev.*, 110:145–172, 2003.
- 52 N. Savva, A. Scarinzi, and N. Berthouze. Continuous recognition of player’s affective body expression as dynamic quality of aesthetic experience. *IEEE Transactions on Computational Intelligence and AI in Games*, 2012.
- 53 K. R. Scherer. Studying the emotion-antecedent appraisal process: An expert system approach. *Cognition and Emotion*, 7:325–355, 1993.
- 54 K.R. Scherer. What are emotions? and how can they be measured? *Social science information*, 44(4):695–729, 2005.
- 55 N. Shaker, S. Asteriadis, G. Yannakakis, and K. Karpouzis. A game-based corpus for analysing the interplay between game context and player experience. In *Affective Computing and Intelligent Interaction*, pages 547–556. Springer, 2011.
- 56 M. Sharma, M. Mehta, S. Ontanón, and A. Ram. Player modeling evaluation for interactive fiction. In *Proceedings of the AIIDE 2007 Workshop on Optimizing Player Satisfaction*, pages 19–24, 2007.
- 57 B. F. Skinner. *The Behavior of Organisms: An Experimental Analysis*. Cambridge, Massachusetts: B. F. Skinner Foundation, 1938.
- 58 A.M. Smith, C. Lewis, K. Hullet, and A. Sullivan. An inclusive view of player modeling. In *Proceedings of the 6th International Conference on Foundations of Digital Games*, pages 301–303. ACM, 2011.
- 59 Nathan Sorenson and Philippe Pasquier. Towards a generic framework for automated video game level creation. In *Proceedings of the European Conference on Applications of Evolutionary Computation (EvoApplications)*, volume 6024, pages 130–139. Springer LNCS, 2010.
- 60 Pieter Spronck, Ida Sprinkhuizen-Kuyper, and Eric Postma. Difficulty Scaling of Game AI. In *Proceedings of the 5th International Conference on Intelligent Games and Simulation (GAME-ON 2004)*, pages 33–37, 2004.
- 61 S. Tekofsky, P. Spronck, A. Plaat, H.J. van den Herik, and J. Broersen. Psyops: Personality assessment through gaming behavior. In *Proceedings of the FDG 2013*, 2013.
- 62 Ruck Thawonmas, Masayoshi Kurashige, Keita Iizuka, and Mehmed Kantardzic. Clustering of Online Game Users Based on Their Trails Using Self-organizing Map. In *Proceedings of Entertainment Computing - ICEC 2006*, pages 366–369, 2006.
- 63 D. Thue, V. Bulitko, M. Spetch, and E. Wasylshen. Learning player preferences to inform delayed authoring. In *AAAI Fall Symposium on Intelligent Narrative Technologies*. AAAI Press, Arlington, 2007.
- 64 David Thue, Vadim Bulitko, Marcia Spetch, and Eric Wasylshen. Interactive storytelling: A player modelling approach. In *The Third Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, pages 43–48, Stanford, CA, 2007.
- 65 Christian Thurau, Christian Bauckhage, and Gerhard Sagerer. Learning human-like Movement Behavior for Computer Games. In S. Schaal, A. Ijspeert, A. Billard, Sethu Vijayakumar, J. Hallam, and J.-A. Meyer, editors, *From Animals to Animats 8: Proceedings of the 8<sup>th</sup> International Conference on Simulation of Adaptive Behavior (SAB-04)*, pages 315–323, Santa Monica, LA, CA, July 2004. The MIT Press.

- 66 Julian Togelius, Mike Preuss, Nicola Beume, Simon Wessing, Johan Hagelbäck, and Georgios N. Yannakakis. Multiobjective exploration of the starcraft map space. In *Proceedings of the IEEE Conference on Computational Intelligence and Games (CIG)*, 2010.
- 67 Julian Togelius, Georgios N. Yannakakis, Kenneth O. Stanley, and Cameron Browne. Search-based procedural content generation. In *Proceedings of EvoApplications*, volume 6024. Springer LNCS, 2010.
- 68 S. Tognetti, M. Garbarino, A. Bonarini, and M. Matteucci. Modeling enjoyment preference from physiological responses in a car racing game. In *Computational Intelligence and Games (CIG), 2010 IEEE Symposium on*, pages 321–328. IEEE, 2010.
- 69 Giel van Lankveld. *Quantifying Individual Player Differences*. PhD thesis, Tilburg University, The Netherlands, 2013.
- 70 Giel van Lankveld, Pieter Spronck, and Matthias Rauterberg. Difficulty Scaling through Incongruity. In *Proceedings of the 4th International Artificial Intelligence and Interactive Digital Entertainment Conference*, pages 228–229. AAAI Press, 2008.
- 71 Giel van Lankveld, Pieter Spronck, Jaap van den Herik, and Arnoud Arntz. Games as personality profiling tools. In *2011 IEEE Conference on Computational Intelligence in Games (CIG'11)*, pages 197–202, 2011.
- 72 Ben Weber and Michael Mateas. A Data Mining Approach to Strategy Prediction. In *IEEE Symposium on Computational Intelligence in Games (CIG 2009)*, pages 140–147, Milan, Italy, September 2009.
- 73 G. Yannakakis and J. Hallam. Game and player feature selection for entertainment capture. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games*, pages 244–251, 2007.
- 74 G. Yannakakis and J. Hallam. Rating vs. preference: a comparative study of self-reporting. *Affective Computing and Intelligent Interaction*, pages 437–446, 2011.
- 75 Georgios N. Yannakakis and John Hallam. Towards Optimizing Entertainment in Computer Games. *Applied Artificial Intelligence*, 21:933–971, 2007.
- 76 Georgios N. Yannakakis and Manolis Maragoudakis. Player modeling impact on player’s entertainment in computer games. In *Proceedings of the 10<sup>th</sup> International Conference on User Modeling; Lecture Notes in Computer Science*, volume 3538, pages 74–78, Edinburgh, 24–30 July 2005. Springer-Verlag.
- 77 G.N. Yannakakis. Preference learning for affective modeling. In *Affective Computing and Intelligent Interaction and Workshops, 2009. ACII 2009. 3rd International Conference on*, pages 1–6. IEEE, 2009.
- 78 G.N. Yannakakis. Game AI Revisited. In *Proceedings of the 9th conference on Computing Frontiers*, pages 285–292. ACM, 2012.
- 79 G.N. Yannakakis, H.P. Martínez, and A. Jhala. Towards affective camera control in games. *User Modeling and User-Adapted Interaction*, 20(4):313–340, 2010.
- 80 G.N. Yannakakis and J. Togelius. Experience-driven procedural content generation. *Affective Computing, IEEE Transactions on*, 2(3):147–161, 2011.
- 81 Z. Zeng, M. Pantic, G.I. Roisman, and T.S. Huang. A survey of affect recognition methods: Audio, visual, and spontaneous expressions. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(1):39–58, 2009.



## Literature review as a research methodology: An overview and guidelines

Hannah Snyder



BI-Norwegian School of Business, Nydalsveien 37, 0484 Oslo, Norway

### ARTICLE INFO

**Keywords:**

Literature review  
Synthesis  
Research methodology  
Systematic review  
Integrative review

### ABSTRACT

Knowledge production within the field of business research is accelerating at a tremendous speed while at the same time remaining fragmented and interdisciplinary. This makes it hard to keep up with state-of-the-art and to be at the forefront of research, as well as to assess the collective evidence in a particular area of business research. This is why the literature review as a research method is more relevant than ever. Traditional literature reviews often lack thoroughness and rigor and are conducted ad hoc, rather than following a specific methodology. Therefore, questions can be raised about the quality and trustworthiness of these types of reviews. This paper discusses literature review as a methodology for conducting research and offers an overview of different types of reviews, as well as some guidelines to how to both conduct and evaluate a literature review paper. It also discusses common pitfalls and how to get literature reviews published.

### 1. Introduction

Building your research on and relating it to existing knowledge is the building block of all academic research activities, regardless of discipline. Therefore, to do so accurately should be a priority for all academics. However, this task has become increasingly complex. Knowledge production within the field of business research is accelerating at a tremendous speed while at the same time remaining fragmented and interdisciplinary. This makes it hard to keep up with state-of-the-art research and to be at the forefront, as well as to assess the collective evidence in a particular research area. This is why the literature review as a research method is more relevant than ever. A literature review can broadly be described as a more or less systematic way of collecting and synthesizing previous research (Baumeister & Leary, 1997; Tranfield, Denyer, & Smart, 2003). An effective and well-conducted review as a research method creates a firm foundation for advancing knowledge and facilitating theory development (Webster & Watson, 2002). By integrating findings and perspectives from many empirical findings, a literature review can address research questions with a power that no single study has.

It can also help to provide an overview of areas in which the research is disparate and interdisciplinary. In addition, a literature review is an excellent way of synthesizing research findings to show evidence on a meta-level and to uncover areas in which more research is needed, which is a critical component of creating theoretical frameworks and building conceptual models. However, traditional ways of describing and portraying the literature often lack thoroughness and are not undertaken systematically (Tranfield et al., 2003). This results in a lack of

knowledge of what the collection of studies is actually saying or to what it is pointing at. As a result, there is a great chance that authors build their research on flawed assumptions. When researchers are selective of the evidence on which to build their research, ignoring research that points the other way, serious problems can be faced. In addition, even when the methodology of the reviews is valid, there are often issues with what constitutes a good contribution.

Of course, there already exist some guidelines for conducting literature reviews that suggest different types of reviews, such as narrative or integrative reviews (e.g., Baumeister & Leary, 1997; Wong, Greenhalgh, Westhorp, Buckingham, & Pawson, 2013), systematic reviews, and meta-analysis (e.g., Davis, Mengersen, Bennett, & Mazerolle, 2014; Liberati et al., 2009; Moher, Liberati, Tetzlaff, & Altman, 2009) or integrative reviews (e.g., Torraco, 2005). There have also been some attempts to develop guidelines specifically for business or management research (e.g., Palmatier, Houston, & Hulland, 2018; Tranfield et al., 2003). By building on and synthesizing these different types of literature reviews, this paper takes a broader view by summarizing and integrating the different guidelines, including how to apply them in business research. More specifically, the purpose of this paper is to provide an overview of and guidelines for different types of literature reviews as a research method in business research.

In the following paper, it will be argued that the potential for making theoretical and practical contributions using the literature review as a method will be advanced by clarifying what a literature review is, how it can be used, and what criteria should be used to evaluate its quality. The paper has several contributions. First, this paper separates between different types of review methodologies; systematic,

E-mail address: [hannah.snyder@bi.no](mailto:hannah.snyder@bi.no).

**Table 1**  
Approaches to literature reviews.

| Approach                 | Systematic                                       | Semi-systematic   | Integrative  |
|--------------------------|--|---|--|
| Typical purpose          | Synthesize and compare evidence                  | Overview research area and track development over time  | Critique and synthesize                                      |
| Research questions       | Specific   | Broad   | Narrow or broad  |
| Search strategy          | Systematic                                       | May or may not be systematic  | Usually not systematic                                       |
| Sample characteristics   | Quantitative articles                            | Research articles   | Research articles, books, and other published texts          |
| Analysis and evaluation  | Quantitative                                     | Qualitative/quantitative  | Qualitative  |
| Examples of contribution | Evidence of effect<br>Inform policy and practice | State of knowledge<br>Themes in literature<br>Historical overview<br>Research agenda<br>Theoretical model | Taxonomy or classification<br>Theoretical model or framework |

semi-systematic and integrative approaches and argues that depending on purpose and the quality of execution, each type of approach can be very effective. While systematic reviews have strict requirements for search strategy and selecting articles for inclusion in the review, they are effective in synthesizing what the collection of studies are showing in a particular question and can provide evidence of effect that can inform policy and practice. However, systematic reviews are not always the best strategy. Instead, when wanting to study a broader topic that has been conceptualized differently and studied within diverse disciplines, this can hinder a full systematic review process. Instead, a semi-systematic review approach could be a good strategy for example map theoretical approaches or themes as well as identifying knowledge gaps within the literature. In some cases, a research question requires a more creative collection of data, in these cases; an integrative review approach can be useful when the purpose of the review is not to cover all articles ever published on the topic but rather to combine perspectives to create new theoretical models. Second, the current paper addresses practical issues that may be encountered when conducting a literature review in business research. These issues, for example, can relate to selecting the appropriate review methodology for the targeted research question, deciding on eligibility criteria, determining appropriate boundaries for the review, choosing what data to extract from the paper, or concluding what type of contribution should be made. Third, this paper provides context and guidance to researchers seeking to use the literature review as a method to synthesize research in their own domains, to inform their own research, or to provide guidance for social policy. Last, this paper also aims to provide some guidelines for how to assess quality when evaluating review papers which hopefully will be helpful to editors, reviewers, and authors, as well as to any reader of a review paper.

## 2. Why you should write a literature review

Consideration of prior, relevant literature is essential for all research disciplines and all research projects. When reading an article, independent of discipline, the author begins by describing previous research to map and assess the research area to motivate the aim of the study and justify the research question and hypotheses. This is generally referred to as the “literature review,” “theoretical framework,” or “research background.” However, for a literature review to become a proper research methodology, as with any other research, follow proper steps need to be followed and action taken to ensure the review is accurate, precise, and trustworthy. As with all research, the value of an academic review depends on what was done, what was found, and the clarity of reporting (Moher et al., 2009). Depending on the purpose of the review, the researcher can use a number of strategies, standards, and guidelines developed especially for conducting a literature review. Then, when should a literature review be used as a research method?

For a number of research questions, a literature review may be the best methodological tool to provide answers. For example, reviews are useful when the researcher wants to evaluate theory or evidence in a

certain area or to examine the validity or accuracy of a certain theory or competing theories (Tranfield et al., 2003). This approach can be narrow, such as investigating the effect of or relationship between two specific variables, or it can be broader, such as exploring the collective evidence in a certain research area. In addition, literature reviews are useful when the aim is to provide an overview of a certain issue or research problem. Typically, this type of literature review is conducted to evaluate the state of knowledge on a particular topic. It can be used, for example, to create research agendas, identify gaps in research, or simply discuss a particular matter. Literature reviews can also be useful if the aim is to engage in theory development (Baumeister & Leary, 1997; Torrao, 2005). In these cases, a literature review provides the basis for building a new conceptual model or theory, and it can be valuable when aiming to map the development of a particular research field over time. However, it is important to note that depending on the goal of the literature review, the method that should be used will vary.

### 2.1. Different approaches to conducting a literature review

As mentioned previously, there are a number of existing guidelines for literature reviews. Depending on the methodology needed to achieve the purpose of the review, all types can be helpful and appropriate to reach a specific goal (for examples, please see Table 1). These approaches can be qualitative, quantitative, or have a mixed design depending on the phase of the review. In the following, three broad types of methods commonly used will be described, as summarized in Table 2. The broad types that will be presented and discussed include the systematic review, the semi-systematic review, and the integrative review. Under the right circumstances, all of these review strategies can be of significant help to answer a particular research question. However, it should be noted that there are many other forms of literature reviews, and elements from different approaches are often combined. As these approaches are quite wide, it should be noted that they might require further adaptation for a particular research project.

#### 2.1.1. Systematic literature review

*What is it and when should we use it?* Systematic reviews have foremost been developed within medical science as a way to synthesize research findings in a systematic, transparent, and reproducible way and have been referred to as the gold standard among reviews (Davis et al., 2014). Despite all the advantages of this method, its use has not been overly prevalent in business research, but it is increasing (e.g., Snyder, Witell, Gustafsson, Fombelle, & Kristensson, 2016; Verlegh & Steenkamp, 1999; Witell, Snyder, Gustafsson, Fombelle, & Kristensson, 2016). A systematic review can be explained as a research method and process for identifying and critically appraising relevant research, as well as for collecting and analyzing data from said research (Liberati et al., 2009). The aim of a systematic review is to identify all empirical evidence that fits the pre-specified inclusion criteria to answer a particular research question or hypothesis. By using explicit and systematic methods when reviewing articles and all available evidence, bias can be

**Table 2**

Examples of existing guidelines for conducting a literature review.

| Authors                     | Discipline      | Type of literature review            | Contribution  |
|-----------------------------|-----------------|--------------------------------------|---|
| Baumeister and Leary (1997) | Psychology      | Narrative review                     | <ul style="list-style-type: none"> <li>Overviews reasons for conducting a review</li> <li>Discusses common mistakes for conducting a review</li> <li>Compares management and healthcare research</li> <li>Highlights the challenges of conducting a systematic review in management research</li> <li>Provides guidelines for conducting a systematic literature review in management research</li> </ul> |
| Tranfield et al. (2003)     | Management      | Systematic review                    | <ul style="list-style-type: none"> <li>Defines the integrative literature review</li> <li>Provides guidelines and examples for integrative literature reviews</li> <li>Discusses contributions of a integrative literature review</li> <li>Provides guidelines for conducting and reporting systematic reviews and meta-analysis</li> </ul>   |
| Torraco (2005)              | Human Resources | Integrative review                   | <ul style="list-style-type: none"> <li>Provides guidelines and examples for integrative literature reviews</li> <li>Discusses contributions of a integrative literature review</li> <li>Provides guidelines for conducting and reporting systematic reviews and meta-analysis</li> </ul>  |
| Liberati et al. (2009)      | Medicine        | Systematic review and meta-analysis  | <ul style="list-style-type: none"> <li>Provides guidelines for conducting a meta-narrative review</li> <li>Synthesizes guidelines for systematic literature reviews</li> <li>Provides guidelines for conducting a systematic review and meta-analysis in social sciences</li> <li>Provides guidelines for publishing review papers in the Journal of the Academy of Marketing Science</li> </ul>          |
| Wong et al. (2013)          | Medicine        | Semi-systematic review               |   |
| Davis et al. (2014)         | Social Sciences | Systematic review and meta-analysis  |   |
| Palmatier et al. (2018)     | Marketing       | Review papers and systematic reviews |   |

minimized, thus providing reliable findings from which conclusions can be drawn and decisions made (Moher et al., 2009).

*What type of analysis can be conducted?* Often, but not always, statistical methods, such as the meta-analysis, are used to integrate the results of the included studies. A meta-analysis is a statistical method of combining results from different studies to weigh and compare and to identify patterns, disagreements, or relationships that appear in the context of multiple studies on the same topic (Davis et al., 2014). With the meta-analysis approach, each primary study is abstracted and coded, and findings are subsequently transformed into a common metric to calculate an overall effect size (Glass, 1976). However, to be able to perform a meta-analysis, the included studies must share statistical measures (effect size) to compare results (DerSimonian & Laird, 1986). Therefore, it is challenging to perform a meta-analysis on studies with different methodological approaches (Tranfield et al., 2003). Even though the systematic review method was developed in medical science, attempts have been made to create guidelines within the social sciences (Davis et al., 2014; Palmatier et al., 2018; Tranfield et al., 2003). In addition, there are several published meta-analyses in higher-ranked business journals (Carrillat, Legoux, & Hadida, 2018; Chang & Taylor, 2016). However, in these areas, which are not restricted to randomized controlled trials, a major challenge lies in assessing the quality of research findings. As a result, more qualitative approaches have been developed to assess the quality and strength of findings from different types of studies and to compare results (Greenhalgh, Robert, Macfarlane, Bate, & Kyriakidou, 2004). This is often referred to as a qualitative systematic review, which can be described as a method of comparing findings from qualitative studies (Grant & Booth, 2009). That is, a strict systematic review process is used to collect articles, and then a qualitative approach is used to assess them.

*What is a potential contribution from a systematic review?* There are several advantages and potential contributions of conducting a systematic review. For example, we can determine whether an effect is constant across studies and discover what future studies are required to be conducted to demonstrate the effect. Techniques can also be used to discover which study-level or sample characteristics have an effect on the phenomenon being studied, such as whether studies conducted in one cultural context show significantly different results from those conducted in other cultural contexts (Davis et al., 2014).

### 2.1.2. Semi-systematic review

*What is it and how should it be used?* The semi-systematic or narrative review approach is designed for topics that have been conceptualized differently and studied by various groups of researchers within diverse disciplines and that hinder a full systematic review process (Wong et al., 2013). That is, to review every single article that could be relevant to

the topic is simply not possible, so a different strategy must be developed. There are several examples of articles using this approach published in business journals (e.g., McColl-Kennedy et al., 2017). Besides the aim of overviewing a topic, a semi-systematic review often looks at how research within a selected field has progressed over time or how a topic has developed across research traditions. In general, the review seeks to identify and understand all potentially relevant research traditions that have implications for the studied topic and to synthesize these using meta-narratives instead of by measuring effect size (Wong et al., 2013). This provides an understanding of complex areas. However, while covering broad topics and different types of studies, this approach holds that the research process should be transparent and should have a developed research strategy that enables readers to assess whether the arguments for the judgments made were reasonable, both for the chosen topic and from a methodological perspective.

*What type of analysis can be conducted?* A number of methods can be used to analyze and synthesize findings from a semi-systematic review. These methods often have similarities to approaches used in qualitative research in general. For example, a thematic or content analysis is a commonly used technique and can be broadly defined as a method for identifying, analyzing, and reporting patterns in the form of themes within a text (Braun & Clarke, 2006). Although this type of review is usually followed by a qualitative analysis, there are exceptions. For example, Borman and Dowling (2008) used a semi-structured method of collecting literature but combined it with a statistical meta-analysis approach.

*What is a potential contribution from a semi-systematic review?* This type of analysis can be useful for detecting themes, theoretical perspectives, or common issues within a specific research discipline or methodology or for identifying components of a theoretical concept (Ward, House, & Hamer, 2009). A potential contribution could be, for example, the ability to map a field of research, synthesize the state of knowledge, and create an agenda for further research or the ability to provide an historical overview or timeline of a specific topic.

### 2.1.3. Integrative review

*What is it and when should it be used?* Closely related to the semi-structured review approach is the integrative or critical review approach. In comparison to the semi-structured review, an integrative review usually has a different purpose, with the aim to assess, critique, and synthesize the literature on a research topic in a way that enables new theoretical frameworks and perspectives to emerge (Torraco, 2005). Although rare, examples of this type of review can be identified in the business literature (e.g., Covington, 2000; Gross, 1998; Mazumdar, Raj, & Sinha, 2005). Most integrative literature reviews are intended to address mature topics or new, emerging topics. In the case

**Table 3**

Important questions to consider in each step of the review.

## Phase 1: design

- Is this review needed and what is the contribution of conducting this review?
- What is the potential audience of this review?
- What is the specific purpose and research question(s) this review will be addressing?
- What is an appropriate method to use of this review's specific purpose?
- What is the search strategy for this specific review? (including search terms, databases, inclusion and exclusion criteria etc.)

## Phase 2: conduct

- Does the search plan developed in phase one work to produce an appropriate sample or does it need adjustment?
- What is the practical plan for selecting articles?
- How will the search process and selection be documented?
- How will the quality of the search process and selection be assessed?

## Phase 3: analysis

- What type of information needs to be abstracted to fulfill the purpose of the specific review?
- What type of information is needed to conduct the specific analysis?
- How will reviewers be trained to ensure the quality of this process?
- How will this process be documented and reported?

## Phase 4: structuring and writing the review

- Are the motivation and the need for this review clearly communicated?
- What standards of reporting are appropriate for this specific review?
- What information needs to be included in the review?
- Is the level of information provided enough and appropriate to allow for transparency so readers can judge the quality of the review?
- The results clearly presented and explained?
- Is the contribution of the review clearly communicated?

of mature topics, the purpose of using an integrative review method is to overview the knowledge base, to critically review and potentially re-conceptualize, and to expand on the theoretical foundation of the specific topic as it develops. For newly emerging topics, the purpose is rather to create initial or preliminary conceptualizations and theoretical models, rather than review old models. This type of review often requires a more creative collection of data, as the purpose is usually not to cover all articles ever published on the topic but rather to combine perspectives and insights from different fields or research traditions.

*What type of analysis can be used?* The data analysis part of an integrative or critical review is not particularly developed according to a specific standard (Whittemore & Knafl, 2005). However, while there is no strict standard, the general aim of a data analysis in an integrative review is to critically analyze and examine the literature and the main ideas and relationships of an issue. It should be noted that this requires researchers to have advanced skills, such as superior conceptual thinking (MacInnis, 2011) at the same time as being transparent and document the process of analysis.

*What is a potential contribution from an integrative review?* An integrative review method should result in the advancement of knowledge and theoretical frameworks, rather than in a simply overview or description of a research area. That is, it should *not* be descriptive or historical but should preferably generate a new conceptual framework or theory. Although an integrative review can be conducted in a number of ways, researchers are still expected to follow accepted conventions for reporting on how the study was conducted (Torraco, 2005). That is, how the integrative was done and how articles were selected must be transparent. However, a note of caution. While well-conducted integrative reviews can make a valid and strong contribution to its field of research, more often than the opposite, they either lack transparency or true integration of research. Frequently, reviews labeled as integrative are simply summaries of studies and not truly integrative.

## 2.2. How to decide on what approach to use

While it can be challenging to determine what approach is most appropriate for a specific type of review, the research question and

specific purpose of the review always determine the right strategy to use. While the systematic review is perhaps the most accurate and rigorous approach to collect articles, because there is certainty that all relevant data have been covered, this approach requires a narrow research question, and it might not be feasible or even suitable for all types of projects. This is where the semi-systematic review can be useful, but this approach is also more problematic and as it has fewer clear steps to follow. While the methodology for systematic reviews is straightforward and follows highly strict rules and standards (Liberati et al., 2009), the semi-systematic review process requires more development and tailoring to the specific project (Wong et al., 2013). Often, researchers need to develop their own standards and a detailed plan to ensure the appropriate literature is accurately covered to be able to answer their research question and be transparent about the process. However, if done properly, this can be a highly effective way of covering more areas and broader topics than a systematic review can handle. In addition, when it comes to the integrative review, it becomes even more demanding, which puts more responsibility on and requires more skills of the researchers, as there are even fewer standards and guidelines on which to rely for developing a strategy (Torraco, 2005). This leads to the notion that an integrative review approach might not be advisable to use, and if compared to the systematic review, it might not hold the same amount of rigor. However, if successfully conducting a truly integrative review and contributing with a new conceptual model or theory, the reward can be significant (MacInnis, 2011).

## 3. The process of conducting a literature review

Independent of what approach will be used to conduct the literature review, a number of steps that must be taken and decisions made to create a review that meets the requirements for publication (for specific considerations in relationship to each step. See Table 3). In the following, the basics steps and important choices involved in conducting a literature review will be suggested and discussed using four phases; (1) designing the review, (2) conducting the review, (3) analysis and (4) writing up the review. This process was developed from practical experience and is a synthesis of and influenced by various standards and guidelines suggested for literature reviews (e.g., Liberati et al., 2009; Tranfield et al., 2003; Wong et al., 2013).

### 3.1. Phase 1: designing the review

The first question that should be asked is why this review should be conducted. Is there really a need for a literature review in this area? If so, what type of literature review would be the most helpful and would make the greatest contribution? Of consideration should also be what audience will most likely be interested in the review when deciding on the topic. This is a relevant question because it determines the likelihood of the review being published and the impact it will have on the research community. Conducting a literature review is hard work, so the topic must be one that is of interest to both the author and reader. Therefore, it is a good idea to scan the area as a first step to account for other literature reviews that already exist, to assess the number of research studies that must be assessed, and to help formulate and clearly define the purpose, scope, and specific research question the review will address. These are important actions because they will help to identify which approach is most appropriate. For example, if the review aims to summarize or evaluate a large field of research or even several research areas, a strict systematic review approach may not be suitable or even possible. Instead, a narrative or integrative review approach would be preferable. In the same way, if the purpose of the review is to investigate and synthesize evidence of the effect of a specific factor, an integrative review is not trustworthy; instead, a systematic review approach should be used. The stated purpose should then guide the rest of the review.

Once the research question has been identified and an overall

review approach considered, a search strategy for identifying relevant literature must be developed. This includes selecting search terms and appropriate databases and deciding on inclusion and exclusion criteria. Here, a number of important decisions must be made that are crucial and will eventually determine the quality and rigor of the review. Search terms can be words or phrases used to access appropriate articles, books, and reports. These terms should be based on words and concepts that are directly related to the research question. Depending on the aim of the review and the research question, these search terms can be broad or narrow. Importantly, it could be worthwhile to consider including additional limitations.

As almost all initial literature searches yield many articles, a strategy is needed to identify which are actually relevant. Inclusion criteria for the review should be guided by the selected research question. Criteria that can be considered and are commonly used are, for example, year of publication, language of the article, type of article (such as conceptual, randomized controlled trial, etc.), and journal. In terms of research quality, deciding on inclusion and exclusion criteria is one of the most important steps when conducting your review. However, important to note is the need to provide reasoning and transparency concerning all choices made; there must be logical and valid motives. This is important, as, independent of the type of approach, the quality of the literature is dependent on, among other aspects, what literature is included and how it was selected (Tranfield et al., 2003; Wong et al., 2013). Depending on these decisions, a study can end up with very different answers and conclusions to the same research questions. For example, by only selecting some specific journals, years, or even search terms to try to limit your search, you can end up with a very flawed or skewed sample and missing studies that would have been relevant to your case or even contradict other studies. You can also come to the wrong conclusion about gaps in the literature, or perhaps more serious, provide false evidence of a specific effect. A practical approach is to write all decisions down to enable transparency, as the authors must be clear in a way that enables the reader to understand how the literature was identified, analyzed, synthesized, and reported. This should be done carefully and prior to actually conducting the review.

### 3.2. Phase 2: conducting the review

After deciding on the purpose, specific research questions, and type of approach, it is time to start conducting the actual review. When conducting the review, a pilot test of the review process and protocol is appropriate. By testing the search terms and inclusion criteria on a smaller sample, the process can be adjusted before performing the main review. It is common to adjust the process a number of times before actually selecting the final sample. Importantly, it should be noted that it is preferred to use two reviewers to select articles to ensure the quality and reliability of the search protocol.

The actual selection of the sample can be done in a number of ways, depending on the nature and scope of the specific review. Depending on how many articles are yielded, different approaches will be appropriate. For example, reviewers may read each piece of literature that appears in the search in full; this is a highly useful, but time-consuming approach. Another option could be to focus on the research method or findings, and a third option is to conduct the review in stages by reading abstracts first and making selections and then reading full-text articles later, before making the final selection. Once this is done and the initial articles (or other relevant literature) have been collected, the texts should be screened in full to ensure they meet the inclusion criteria. As an additional strategy, references in the selected articles can be scanned to identify other articles that may potentially be relevant (however, this is not appropriate when using the systematic review method as this requires a more strict protocol). During this time, the process of including and excluding specific articles should be documented carefully.

### 3.3. Phase 3: analysis

After conducting the literature review and deciding on a final sample, it is important to consider how the articles will be used to conduct an appropriate analysis. That is, after selecting a final sample, a standardized means of abstracting appropriate information from each article should be used. Data abstracted can be in the form of descriptive information, such as authors, years published, topic, or type of study, or in the form of effects and findings. It can also take the form of conceptualizations of a certain idea or theoretical perspective. Importantly, this should be done in concordance with the purpose and research question of the specific review, and the form will vary. In this step, it is important to consider training the reviewers to avoid any differences in coding and abstraction (if there is more than one) and monitoring the data abstraction carefully during the review process to ensure quality and reliability. Often, if the aim is to publish in an academic journal, this will require a detailed description of the process or a measure of reliability between reviewers. Sometimes this is easy, if the information of interest is, for example, population, effect size, or sample size. However, it becomes harder when the information of interest is themes in the literature, perspectives, or providing an historical timeline.

Depending on the review, different analysis methods can be used and are more or less appropriate (please see above for different contributions from different approaches). Nevertheless, independent of the method of analysis, it is important to ensure that it is appropriate to answer the selected research question. For example, if the purpose is to evaluate evidence of the effect of loyalty programs, the use of a meta-analysis is most appropriate. On the other hand, if the purpose was to develop a theoretical model or framework for customer experience, a strict meta-analysis would be a poor choice; rather, an analysis technique suitable for integrative reviews should be used.

### 3.4. Phase 4: writing the review

First, when writing the review, the motivation and need for the review must be clearly communicated. Depending on the approach, the final review article can be structured in different ways, and it will require different types of information and different levels of detail. A number of standards and guidelines explicitly address how literature reviews should be reported and structured, including PRISMA, developed for systematic literature reviews and meta-analyses (see Liberati et al., 2009); RAMSES, developed for systematic narrative reviews (see Wong et al., 2013); and guidelines for integrative reviews (Torrao, 2005). Although review articles can be organized in various ways, some generalizations can be made. All authors are expected to follow accepted conventions for reporting on how the study was undertaken. It is necessary to describe transparently the process of designing the review and the method for collecting literature, that is, how the literature was identified, analyzed, synthesized, and reported by the author. Doing so properly gives the reader the chance to assess the quality and trustworthiness of the findings. The contribution of the specific literature review can take a number of forms, and it should be judged in relationship to the field to which it wants to contribute. Depending on a number of factors, such as the maturity of the field or state of knowledge, different contributions could be valuable. For example, literature reviews can result in a historical analysis of the development within a research field (e.g. Carlborg, Kindström, & Kowalkowski, 2014), an agenda for further research (e.g., McColl-Kennedy et al., 2017), a conceptual model or categorization (e.g., Snyder et al., 2016; Witell et al., 2016), or evidence of an effect (e.g., Verlegh & Steenkamp, 1999).

## 4. Assessing the quality of a literature review

Literature reviews need to be assessed and evaluated as strictly as empirical articles, but is this always the case? Palmatier et al. (2018)

**Table 4**

Guidelines to assess the quality of a literature review.

## Phase 1: design

- Is relationship to the overall research field, is this literature review needed and does it make a substantial, practical, or theoretical contribution?
- Are the motivation, the purpose, and the research question(s) clearly stated and motivated?
- Does the review account for the previous literature review and other relevant literature?
- Is the approach/methodology for the literature review clearly stated?
- Is this the most appropriate approach to address the research problem?
- Are the methodology and the search strategy clearly and transparently described and motivated (including search terms, databases used, and explicit inclusion and exclusion criteria)?

## Phase 2: conduct

- Is the search process appropriate for this type of review?
- Is the practical search process accurately described and accounted for?
- Is the process of the inclusion and exclusion of articles transparent?
- Have proper measures been taken to ensure research quality?
- Can it be trusted that the final sample is appropriate and in concordance with the overall purpose of the review?

## Phase 3: data abstraction and analysis

- Is the data abstracted from the article appropriate in concordance with the overall purpose of the review?
- Is the process for abstracting data accurately described?
- Have proper measures been taken to ensure quality data abstraction?
- Is the chosen data analysis technique appropriate in relation to the overall research question and the data abstracted?
- Is the analysis process properly described and transparent?

## Phase 4: structuring and writing the review

- Is the review article organized coherently in relation to the overall approach and research question?
- Is the overall method of conducting the literature review sufficiently described? Can the study be replicated?
- Is the result of the review reported in an appropriate and clear way?
- Does the article synthesize the findings of the literature review into a clear and valuable contribution to the topic?
- Are questions or directions for further research included? Are the results from the review useable?

suggest that a quality literature review must have both *depth* and *rigor*, that is, it needs to demonstrate an appropriate strategy for selecting articles and capturing data and insights and to offer something beyond a recitation of previous research. In addition, they state that a quality literature review needs to be *replicable*, that is, the method must be described such that an external reader could replicate the study and reach similar findings. Lastly, they state that a literature review must be *useful* for scholars and practitioners. However, evaluating different types of literature reviews can be challenging. Therefore, some guidelines for eventuating literature review articles across approaches are suggested as a starting point to help editors, reviewers, authors, and readers evaluating literature reviews (summarized in Table 4). These depart from the different stages of conducting a literature review and should be broad enough to encompass most types of literature reviews. However, of importance is that when evaluating an individual review, specific standards for the type of review must be examined to assess whether the review meets the criteria for rigor and depth. Depending on if, it is a systematic, semi-systematic or integrative review, different standards can be valid. However, independent of type of review, pay close attention to what studies have been included and for what reasons as these decisions make all the differences in terms of what type of conclusions the authors reach. Ignoring a relevant field of research, some journals or years can have major consequences for the results and conclusions of the studies. In addition, its contribution should always be evaluated against the topic or field to which it adds. What constitutes a useful contribution in one area may be insufficient in another.

## 5. How to get your literature review published

While there are many arguments in favor for conducting a literature review, publishing it can be challenging. Several common mistakes are

made by researchers when conducting a literature review that can hinder it from getting anywhere near publication in a decent journal. First, researchers often fail to describe in enough detail how the literature review was conducted, which makes it impossible to evaluate both the quality of the review and its contribution. These reviews often fail to provide details of the overall research strategy, the selection and exclusion of articles, the limitations of the search method, and the quality of the search process, and they often lack details on how the analysis was conducted. Second, in the eagerness to pare the sample size down to make the review easier to handle, it is also common to limit the search too greatly. This can be done by only including a limited number of journals and a narrow year span or excluding articles from related fields that could have been relevant for the specific review. Limiting the sample too greatly is a warning flag, as it affects both the depth and rigor of the review, and it can have serious effects on its results and contributions. A better way to handle too many samples is to re-consider and narrow down the research question. Of course, sometimes it can be perfectly fine to limit the sample in different ways, but good reasons for doing so must be provided. Third, frequently, researchers who have conducted a review often fail to present and explain the results of the review clearly. Often, a large number of different graphs, tables, and figures is included, but not remarked on or explained. This makes it challenging to understand what they really mean or what was actually found. It is common to spend much time explaining the method and the specific analytical technique, but to spend less time discussing and explaining what was actually found and what these results mean. This is true for both advanced quantitative and qualitative analyses. Failing to put the results in context makes it problematic to judge the contributions of the article.

Lastly, perhaps the most common mistake is that literature reviews often fail to provide a truly valuable contribution to the field. No matter how excellent and rigorous the review article, if it does not provide enough of a contribution, something that is new, it will not be published. Too often, literature reviews are simply descriptive summaries of research conducted between certain years, describing such information as the number of articles published, topics covered, citations analyzed, authors represented, and perhaps methods used, without conducting any deeper analysis. While there are likely times when this can be valuable, this is not usually the case and they are not likely to be published in any journal. In truth, review articles that are a medley of word clouds and citation analyses are highly unlikely to be published. This is a pity, as these researchers have gone through the tedious work of collecting many articles without actually analyzing them in any meaningful way, and because of this, they fail to make a significant contribution.

Yet, there are ways of moving beyond simply summarizing the literature and truly developing something that is new and valuable and create a substantial contribution to the field in question. First, there is of course a need to use a good research methodology that fills the quality criteria for conducting literature reviews, but features or analyses can also be added to make the review paper more likely to stand out. There are many examples of articles that have been successfully published in higher-ranked business journals using a literature review strategy as a basis. Not accounting for the quality of the review itself, there seems to be a number of ways forward. One such way is to conduct a literature review and combine it with a meta-analysis of a relevant topic to provide some evidence of effect. This strategy has been used effectively in articles published in higher-ranked journals (e.g., Carrilat et al., 2018; Edeling & Himme, 2018; Verlegh & Steenkamp, 1999). Important to note is that simply conducting a meta-analysis does not warrant publication; it must also focus on a topic that is relevant, is interesting, and solves some type of research dilemma, thereby advancing the knowledge in the field. In addition, reviews that build on computer-based text analysis and machine learning have been receiving increased interest in business research (e.g., Antons & Breidbach, 2018; Witell et al., 2016). While text analysis may well be an excellent way to

contribute, simply using these types of techniques is not enough if it is not done properly and with a purpose in mind. Often, text analysis approaches end up being highly descriptive, only providing an overview of topics, themes, or networks and not generating any deeper analysis. As an alternative, a valuable output from these types of analyses can for example creating a timeline for analyzing and predicting where a field is heading, a comparison of different related terms or constructs that can serve as a base for theory development or identifying true knowledge gaps in previous research.

Although rare, still highly desirable is a well-executed literature review that provides a new theory or includes a well-grounded substantial research agenda or propositions on which other researchers can build to advance the field (e.g., Boyd & Solarino, 2016; Mazumdar et al., 2005; Rodell, Breitsohl, Schröder, & Keating, 2016). While this type of analysis is often time-consuming and requires strong analytical skills from the researchers, if successful, it can make a great contribution to the specific field of research.

Note that the examples above are only a few of many pathways to making a contribution using literature review as a research method. However, it is very challenging to try to create a contribution when you have already collected the data (published material) and try to turn this into a publishable article. Therefore, it is important to have a particular research question in mind from the beginning and to ensure the right approach is chosen to solve the research problem at hand.

## 6. Summary and conclusion

Literature reviews play an important role as a foundation for all types of research. They can serve as a basis for knowledge development, create guidelines for policy and practice, provide evidence of an effect, and, if well conducted, have the capacity to engender new ideas and directions for a particular field. As such, they serve as the grounds for future research and theory. However, both conducting a literature review and evaluating its quality can be challenging, which is why this paper offers some simple guidelines on how to conduct better, more rigorous literature reviews and, in the long run, simply better research. If there is certainty that the research is built on great accuracy, it will be much easier to identify actual research gaps instead of simply conducting the same research repeatedly, to develop better and more precise hypotheses and research questions, and, therefore, to increase the quality of research as a community.

## References

- Antons, D., & Breidbach, C. F. (2018). Big data, big insights? Advancing service innovation and design with machine learning. *Journal of Service Research*, 21, 17–39. <https://doi.org/10.1177/1094670517738373>.
- Baumeister, R. F., & Leary, M. R. (1997). Writing narrative literature reviews. *Review of General Psychology*, 1, 311–320. <https://doi.org/10.1037/1089-2680.1.3.311>.
- Borman, G. D., & Dowling, N. M. (2008). Teacher attrition and retention: A meta-analytic and narrative review of the research. *Review of Educational Research*, 78, 367–409. <https://doi.org/10.3102/0034654308321455>.
- Boyd, B. K., & Solarino, A. M. (2016). Ownership of corporations: A review, synthesis, and research agenda. *Journal of Management*, 42, 1282–1314. <https://doi.org/10.1177/0149206316633746>.
- Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative Research in Psychology*, 3, 77–101. <https://doi.org/10.1191/1478088706qp063oa>.
- Carlborg, P., Kindström, D., & Kowalkowski, C. (2014). The evolution of service innovation research: A critical review and synthesis. *The Service Industries Journal*, 34(5), 373–398. <https://doi.org/10.1080/02642069.2013.780044>.
- Carrillat, F. A., Legoux, R., & Hadida, A. L. (2018). Debates and assumptions about motion picture performance: A meta-analysis. *Journal of the Academy of Marketing Science*, 46, 273–299. <https://doi.org/10.1007/s11747-017-0561-6>.
- Chang, W., & Taylor, S. A. (2016). The effectiveness of customer participation in new product development: A meta-analysis. *Journal of Marketing*, 80, 47–64. <https://doi.org/10.1509/jm.14.0057>.
- Covington, M. V. (2000). Goal theory, motivation, and school achievement: An integrative review. *Annual Review of Psychology*, 51, 171–200. <https://doi.org/10.1146/annurev.psych.51.1.171>.
- Davis, J., Mengersen, K., Bennett, S., & Mazerolle, L. (2014). Viewing systematic reviews and meta-analysis in social research through different lenses. *SpringerPlus*, 3, 511. <https://doi.org/10.1186/2193-1801-3-511>.
- DerSimonian, R., & Laird, N. (1986). Meta-analysis in clinical trials. *Controlled Clinical Trials*, 7, 177–188. [https://doi.org/10.1016/0197-2456\(86\)90046-2](https://doi.org/10.1016/0197-2456(86)90046-2).
- Edeling, A., & Himme, A. (2018). When does market share matter? New empirical generalizations from a meta-analysis of the market share–performance relationship. *Journal of Marketing*, 82, 1–24. <https://doi.org/10.1509/jm.16.0250>.
- Glass, G. V. (1976). Primary, secondary, and meta-analysis of research. *Educational Researcher*, 5, 3–8. <https://doi.org/10.2307/1174772>.
- Grant, M. J., & Booth, A. (2009). A typology of reviews: An analysis of 14 review types and associated methodologies. *Health Information & Libraries Journal*, 26, 91–108. <https://doi.org/10.1111/j.1471-1842.2009.00848.x>.
- Greenhalgh, T., Robert, G., Macfarlane, F., Bate, P., & Kyriakidou, O. (2004). Diffusion of innovations in service organizations: Systematic review and recommendations. *Milbank Quarterly*, 82, 581–629. <https://doi.org/10.1111/j.0887-378X.2004.00325.x>.
- Gross, J. J. (1998). The emerging field of emotion regulation: An integrative review. *Review of General Psychology*, 2, 271–299. <https://doi.org/10.1037/1089-2680.2.3.271>.
- Liberati, A., Altman, D. G., Tetzlaff, J., Mulrow, C., Götzsche, P. C., Ioannidis, J. P. A., ... Moher, D. (2009). The PRISMA statement for reporting systematic reviews and meta-analyses of studies that evaluate health care interventions: Explanation and elaboration. *Annals of Internal Medicine*, 151, W–65. <https://doi.org/10.7326/0003-4819-151-4-200908180-00136>.
- MacInnis, D. J. (2011). A framework for conceptual contributions in marketing. *Journal of Marketing*, 75, 136–154. <https://doi.org/10.1509/jmkg.75.4.136>.
- Mazumdar, T., Raj, S. P., & Sinha, I. (2005). Reference price research: Review and propositions. *Journal of Marketing*, 69, 84–102. <https://doi.org/10.1509/jmkg.2005.69.4.84>.
- McColl-Kennedy, J. R., Snyder, H., Elg, M., Witell, L., Helkkula, A., Hogan, S. J., & Anderson, L. (2017). The changing role of the health care customer: Review, synthesis and research agenda. *Journal of Service Management*, 28.
- Moher, D., Liberati, A., Tetzlaff, J., & Altman, D. G. (2009). Preferred reporting items for systematic reviews and meta-analyses: The PRISMA statement. *Annals of Internal Medicine*, 151, 264–269. <https://doi.org/10.7326/0003-4819-151-4-200908180-00135>.
- Palmatier, R. W., Houston, M. B., & Hulland, J. (2018). Review articles: Purpose, process, and structure. *Journal of the Academy of Marketing Science*, 46, 1–5. <https://doi.org/10.1007/s11747-017-0563-4>.
- Rodell, J. B., Breitsohl, H., Schröder, M., & Keating, D. J. (2016). Employee volunteering: A review and framework for future research. *Journal of Management*, 42, 55–84. <https://doi.org/10.1177/0149206315614374>.
- Snyder, H., Witell, L., Gustafsson, A., Fombelle, P., & Kristensson, P. (2016). Identifying categories of service innovation: A review and synthesis of the literature. *Journal of Business Research*, 69, 2401–2408. <https://doi.org/10.1016/j.jbusres.2016.01.009>.
- Torraco, R. J. (2005). Writing integrative literature reviews: Guidelines and examples. *Human Resource Development Review*, 4, 356–367. <https://doi.org/10.1177/153448430527283>.
- Tranfield, D., Denyer, D., & Smart, P. (2003). Towards a methodology for developing evidence-informed management knowledge by means of systematic review. *British Journal of Management*, 14, 207–222. <https://doi.org/10.1111/1467-8551.00375>.
- Verlegh, P. W. J., & Steenkamp, J.-B. E. M. (1999). A review and meta-analysis of country-of-origin research. *Journal of Economic Psychology*, 20, 521–546. [https://doi.org/10.1016/S0167-4870\(99\)00023-9](https://doi.org/10.1016/S0167-4870(99)00023-9).
- Ward, V., House, A., & Hamer, S. (2009). Developing a framework for transferring knowledge into action: A thematic analysis of the literature. *Journal of Health Services Research and Policy*, 14, 156–164. <https://doi.org/10.1258/jhsrp.2009.008120>.
- Webster, J., & Watson, R. T. (2002). Analyzing the past to prepare for the future: Writing a literature review. *Management Information Systems Quarterly*, 26, 3.
- Whittemore, R., & Knafl, K. (2005). The integrative review: Updated methodology. *Journal of Advanced Nursing*, 52, 546–553. <https://doi.org/10.1111/j.1365-2648.2005.03621.x>.
- Witell, L., Snyder, H., Gustafsson, A., Fombelle, P., & Kristensson, P. (2016). Defining service innovation: A review and synthesis. *Journal of Business Research*, 69, 2863–2872.
- Wong, G., Greenhalgh, T., Westhorp, G., Buckingham, J., & Pawson, R. (2013). RAMESSES publication standards: Meta-narrative reviews. *BMC Medicine*, 11, 20. <https://doi.org/10.1186/1741-7015-11-20>.
- Hannah Snyder** is an assistant professor at the department of marketing, BI - Norwegian School of Business, Oslo, Norway. Her research interest relates to service innovation, customer creativity, deviant customer behavior, and value co-creation as well as a special interest in literature review methodology. She has published in the *Journal of Business Research*, *European Journal of Marketing*, *Journal of Service Management* and *International Journal of Nursing Studies*.



# How to Write a Literature Review Paper?

Bert Van Wee & David Banister

To cite this article: Bert Van Wee & David Banister (2016) How to Write a Literature Review Paper?, *Transport Reviews*, 36:2, 278-288, DOI: [10.1080/01441647.2015.1065456](https://doi.org/10.1080/01441647.2015.1065456)

To link to this article: <https://doi.org/10.1080/01441647.2015.1065456>



Published online: 10 Jul 2015.



Submit your article to this journal 



Article views: 51023



View related articles 



View Crossmark data 



Citing articles: 57 View citing articles 

## How to Write a Literature Review Paper?

BERT VAN WEE<sup>\*§</sup> AND DAVID BANISTER<sup>\*\*</sup>

<sup>\*</sup>*Faculty of Technology, Policy and Management, Delft University of Technology, Jaffalaan 5, 2628 BX Delft, The Netherlands;* <sup>\*\*</sup>*Transport Studies Unit, School of Geography and the Environment, Oxford University, South Parks Road, Oxford OX1 3QY, UK*

(Received 17 April 2015; revised 16 June 2015; accepted 19 June 2015)

**ABSTRACT** This paper discusses the question about how to write a literature review paper (LRP). It stresses the primary importance of adding value, rather than only providing an overview, and it then discusses some of the reasons for (or not) actually writing an LRP, including issues relating to the nature and scope of the paper. It also presents different types of LRPs, advises on reporting the methodology used for the selection of papers for review, and the structure of an LRP. An important conclusion is that the heterogeneity in LRPs is very large. This paper also presents some of the aspects that the authors feel are important structural and contextual considerations that help produce high-quality review papers.

### 1. Introduction

Literature review papers (LRPs) are often very helpful for researchers, as the reader gets an up-to-date and well-structured overview of the literature in a specific area, and the review adds value. This added value can, for example, be that the research gaps are made explicit, and this may be very helpful for readers who plan to do research in the same area for the first time. Alternatively, the review can outline the advantages and disadvantages of the methods used and the implications of the findings are discussed. This can be very helpful for the reader who needs to interpret and use the findings. A review can also help to refresh the information base of a researcher returning to a subject area after some time away from it. The basic question covered in this paper is about how to carry out an LRP and to illustrate this with examples from transport.

Writing an LRP is much less straightforward than writing a mainstream research paper, as many choices with respect to the structure need to be made. Therefore, some conceptual and methodological guidance for researchers planning to write an LRP would be helpful. But to the best of our knowledge there is no academic paper in the transport literature that takes the aspiring writer through the thought processes surrounding the issues about how to write an LRP. This paper aims to fill this gap.

There is a Transportation Research Board paper giving some guidance, especially for literature reviews as part of wider projects (Avni et al., 2015), and

---

<sup>§</sup>Corresponding author. Email: [g.p.vanwee@tudelft.nl](mailto:g.p.vanwee@tudelft.nl)

the paper also refers to tutorials (videos and websites). And there are papers giving guidance for LRP<sub>s</sub> in other areas. For example, Webster and Watson (2002) discuss literature review in the information systems area, and Denney and Tewksbury (2013) in the area of criminal justice. This paper aims to fill this gap. In the medical literature, many papers publish empirical results for tests of medicine, prescribing protocols for data collections (e.g. double blind), and analyses (standard methods), but in the field of transport a much larger variety of methods is applied, making LRP reviewing methods very different. In the area of engineering and physics, the stochastic component of research is much less important than in social sciences (and often absent), reducing the importance of related aspects in empirical studies. What is very important in social sciences in general, but certainly also in the field of transport, is the fact that many variables influence an independent variable (e.g. travel behaviour) in a complex way, resulting in complex causal relationships, and a multitude of data analysis methods and interpretations.

In addition to LRP<sub>s</sub>, there are many empirical papers that review the literature, but we do not discuss how this should be done in case of empirical studies and limit ourselves to LRP<sub>s</sub>. We define an LRP as a journal paper that provides a comprehensive overview of (or a selection of) the literature in a specific area, bringing together the material in a clearly structured way, and adding value through coming to some interesting conclusions. Our focus is on the more general LRP<sub>s</sub>, excluding the specifics of more quantitative methods, such as meta-analyses (see below) or scientometric analyses (see Van Meeteren, Poorthuis, Derudder, & Witlox, in press, for an example).

The approach used in this paper is the ‘learning by doing’ method. ‘Doing’ in our case is the combination of writing LRP<sub>s</sub> ourselves (often with co-authors), reviewing such papers, teaching Ph.D. students how to write such papers, deciding as editors on the suitability of papers for publication (based on external review reports), and discussing the topic with editorial board members of transport journals, and other academics. The paper aims to provide help for researchers interested in writing an LRP, but we do not provide a template. Rather, we discuss a list of topics that we hope is relevant and helpful. As there are many types of LRP<sub>s</sub> and many ways to structure high-quality reviews, it makes no sense to present a ‘one size fits all’ solution. We limit the paper to writing an LRP for an academic journal in the transport domain. Nevertheless, some of the content may be relevant for other purposes, such as discussing the literature as part of a Ph.D. thesis.

Section 2 explains why to (not) write an LRP, and it includes a discussion on the different means by which the issue of added value can be addressed in writing an LRP. Section 3 presents some examples of types of LRP, and Section 4 outlines the means by which different methodologies can be used to select papers for inclusion in the review. Section 5 gives some guidance on the different means to structure the LRP. Section 6 emphasises the important issue of choice of journal to submit an LRP to, and some of the reasons why many review papers are rejected.

## **2. The Rationale for Writing an LRP and Added Value**

There are many reasons for thinking about writing a review paper, but it must have a clear rationale and the key issue of added value needs to be the central concern throughout the paper. In terms of the rationale, writing a literature review implies a wide range of reading, resulting in the researcher acquiring a

substantial amount of knowledge in the research area, and this on its own might generate the enthusiasm to write an LRP. As a consequence of this, the paper may become heavily cited and this will build the reputation of the author(s), and help in promoting their standing as a learned scholar in the field. This in turn allows researchers to position their own research clearly in the academic literature. As part of their job, many academics would read a substantial range of the literature anyway, and so writing an LRP implies one can 'harvest' all the reading in an explicit way, and get credit for doing it. It is also one of the research activities that can be carried out independently, as one only needs access to the books, the journal articles, and other literature. For example, there is no need to plan for data collection (e.g. questionnaires). Therefore, writing an LRP is to a large extent a 'stand-alone' activity, and it is one form of research output that does in some cases (but definitively not all cases) get well cited. For example, in *Transport Reviews* (since 1994), some papers have received more than 200 citations through the Web of Science (Goodwin, Dargay & Hanley, 2004; Pucher & Buehler, 2008; Yang & Bell, 1998). Literature reviews are also used for teaching, as well as research purposes. For example, we are aware of several teachers who use the Geurs and van Wee (2004) paper for an introduction to the topic of accessibility.

Apart from the positive messages given above about the reasons for wanting to write an LRP, there may also be reasons for not even thinking about the possibility. The most obvious reason might be that the LRP a researcher intends to write has already been published, and if it has the same potential scope as the author has in mind, a new review would be redundant. However, in many cases there might still be potential for an additional review, if a different literature is to be used, or if an existing review needs updating, or even if the same literature is used, but a new angle is being taken resulting in substantially different conclusions. For example, an already published review might have an empirical focus, whereas the new review might have more of a methodological focus. The topic of residential self-selection provides a nice example: Mokhtarian and Cao (2008) review the literature from a methodological angle, whereas Cao, Mokhtarian, and Handy (2009) review this literature from an empirical angle. If the review was carried out some time ago, and since then new and 'better' methodologies have been applied, then there may be a case for a new review, focusing on the recent literature. Alternatively, there may not be enough papers to include in the review, and this may be a reason for not writing an LRP, but the solution here would be to expand the scope of the review. The opposite is more likely, where there are too many papers, and then narrowing down the scope might be the solution. Writing a highly regarded LRP is not an insignificant task, and it is often a time-consuming activity. Not only does the selection and reading of literature in many cases take a considerable time, but so does the writing, as an LRP is much less straightforward than writing a mainstream research paper, and consequently the writing stage might take a considerable time.

When examining research outputs, the review paper is often given less weight than a more traditional paper, as it might not have as much 'original research' as the traditional paper, but this possibility should not reduce its value. It could be argued that a high-quality review paper is of more value to the research community than a high-quality research paper, written within a more conventional research structure and on a more focused topic. As already stated, this may in part be a consequence of the fact that there is no standard template for LRPs, and this might also deter potential authors from writing an LRP. The perception

that a review paper is of less value than a research paper (Steward, 2004) may also result from the confusion between the general overview paper and the critical review paper.

The crucial difference between the more general overview paper and the more critical review paper is central to the issue of added value (Table 1): an overview paper does not need to add value, but review paper does. The many options include the full range of paper types, ranging from conceptual, theoretical, and methodological, to more case study and practice-based reviews. The options can be synthetic, bringing together of different approaches, they can be critical in terms of the review, and they can be innovative in terms of proposing new conceptual frameworks.

Taking the option of empirical insights as an example, the results can be presented in many ways, for example, in the form of a range of quantitative effects of an independent variable on a dependent variable. Meta-analyses are a more rigorous way to present empirical results, because these do not only provide such a range, but also insights into the quantitative importance of influencing factors. For example, Brons, Nijkamp, Pels, and Rietveld (2008) present a meta-analysis of

**Table 1.** Options for the added value of LRP

| Options for added value                  | Comments  | Main output (examples)  |
|--|---|---|
| Empirical insights                       | A synthesis of what is already known (and maybe what is not)  | State of knowledge<br>Gaps in literature<br>Weaknesses of methodologies used  |
| Methodologies                            | An analysis of methods used, and their advantages and disadvantages   | Overview of dominant methodologies used<br>Pros and cons of methodologies used  |
| Theories                                 | An investigation of different theories used, and their importance. This might cover the implications for the results  | Opportunities for new methods<br>Overview of main theories used<br>Strengths and weaknesses<br>Impact of theories used on results<br>Potential for other theories   |
| Gaps in literature and a research agenda | This can relate to reviews with an empirical, methodological, and theoretical focus — to explore omissions and limitations in approaches and suggest ways forward | Main gaps in literature<br>Avenues for future research  |
| Relevance for real-world applications    | A discussion or synthesis of how useful the literature is for real-world applications (policy, planning, etc.) — perhaps with the use of case studies             | Overview of knowledge available for real-world applications<br>Design guidance<br>Examples of real-world cases that are (not) underpinned by results from literature<br>Comparison between cases or countries |
| Conceptual model                         | Provides explicit structure on how dependent and independent variables are related. Can be presented preceding or following the review part of a paper            | Scheme, figure presenting the conceptual model<br>Overview of which parts are (not) well founded/underpinned by literature  |

gasoline price elasticities, including as explanatory factors short-term versus long-term focus, geographic area, year of the study, data type, time horizon, and the functional specification of the demand equation. Another example: reviews can aim to be relevant for real-world applications. For example, Givoni (2006) reviews the literature on high-speed rail, disentangling components of high-speed rail operations, and translating these into design guidance.

A sign of a good review is that the value added permeates the whole review and not just the conclusions. For example, Schwanen's review (2013, p. 232) states,

Thinking about sociotechnical transitions comes, however, in many varieties. One could mobilise, for instance, evolutionary economic theory (Dosi, 1982; Nelson & Winter, 1982); long-wave economic theory (Freeman & Louca, 2001); the multilevel perspective and affiliated approaches (Geels, 2011; Nill & Kemp, 2009); practice theory (Shove & Walker, 2010); and sociologies of complexity (Urry, 2011). Nonetheless, whilst expanding rapidly, applications of transition thinking in transport and mobilities research are as yet fairly limited in number and tend to be animated by the multi-level perspective, practice theory and sociologies of complexity.

This quote demonstrates both a wide-ranging knowledge about several different literatures, and it imposes a clear structure on their usefulness in thinking about sociotechnical transitions.

The key lessons from this section relate to the scope of the review and the added value. The scope needs to take a balance between specificity and generality, as being too specific restricts the range of literature that can be covered and being too general makes it much harder to produce a high-quality review, as there is so much material available. The key here is to have a clear focus to the review, as it is easier to 'grow' a review by extending its scope, rather than trying to 'restrict' a review as the scope is already too large.

In terms of the added value, an LRP needs to have a clear message and interpretation, and this should indeed be a central part of the rationale for writing the paper in the first place. Perhaps it is best to take a problem that can be specified as a series of objectives, and then to structure the review around these. This type of review is very much evidence-based and is similar to a conventional paper. Alternatively, there is also the potential to use a more heuristic approach and leave the objectives more open-ended. This might provide a more appealing approach, but it means that the clear conclusions need to be drawn at the end. In both cases, the interpretation of the material used is central to a high-quality LRP, and a paper with weak or no conclusions must be avoided. Some of these issues are now discussed in more detail.

### **3. Types of LRPs**

Building on the typology outlined in Table 1, LRPs come in many different forms. A classic LRP would first outline the structure and purpose of the review, and it would then present the literature in a logical way, commenting on the differences and similarities between the materials cited, and this would then be followed by discussion and conclusions — it is this last part that relates to the added value of the LRP. But there are many more types. In some cases, a paper can have an

empirical question, and the method adopted in the review is to answer the question by reviewing the literature. A literature review is then used to answer these questions. An alternative is to take a new or non-conventional approach to a well-known problem — this can be important for many reasons, such as to shed new light on an existing topic, to disentangle concepts in subcomponents, or to put the results in another perspective. An example is provided by Geurs and Van Wee (2004), which reviews the literature on accessibility measures. Instead of taking the standard perspective of categories of accessibility measures, they examine the components that contribute to the accessibility measures (land use, transport, the temporal, and the individual component). Another alternative is a paper that does not aim to review all (main) literature in an area, but to cover a specific theme. For example, Banister, Anderton, Bonilla, Givoni, and Schwanen (2011) review on transport and the environment had a clear focus on low-carbon transport systems, behavioural and technological options, demand reduction, and the role of international agreements. But the real core of the review was on rethinking governance with respect to low-carbon transport systems and the means to implement policy change within a fragmented decision-making process. This example illustrates how an under-researched area can be identified for further investigation, even though it is embedded within a well-covered research area. It provides a starting point for new research. A final alternative might be to present a conceptual model and then to explore the literature that might help support such an innovative framework. As for theme papers, not all (main) literature then needs to be reviewed, but the references discussed serve the purpose of underpinning the conceptual model. Van Acker, Van Wee, and Witlox (2010) adopted this approach in their study of travel behaviour by introducing a new framework at an early stage in their paper, and then review the literature on travel behaviour from this perspective. The heterogeneity of types of LRP s all contribute to the fact that writing an LRP is not straightforward, but interesting, challenging, and rewarding.

#### **4. Methodology: Selection of Papers**

One of the weakest elements in LRP s is that they are not explicit in the methodologies used. The issue here is different to the conventional paper, where there is often a section in the paper devoted to the methods that will be used, and comments are then made at the end of the paper on the strengths and weaknesses of the methods used. In LRP s, the section on methods is often very short or not present at all, as the literature used in the review is ‘drawn’ from the extensive publications available. There are different ways to address this limitation and our strong recommendation is for authors to be explicit on the methodologies being used and the selection of the material that forms the source material for the review. In case a paper aims to review more or less all main literature in an area, the most obvious sources are the numerous databases that are widely available (e.g. Web of Science, SCOPUS, Scholar Google, and TRID), and information needs to be given as to how these have been systematically ‘searched’. For example, comments would need to cover the key words used for the search (including strings, such as ‘transport\*’ to include both transport and transportation), and if the selection has been heavily influenced by the Boolean operators (AND, OR, and NOT). In all cases, we recommend making the use of these operators explicit. An excellent example of making explicit the search strategy is the

LRP produced by Scheepers et al. (2014), where they explicitly report on databases, languages included, keywords, search strategy, and some other aspects. In addition, the languages covered should be made explicit, especially if literature in other languages than English is covered. The time frame should also be made explicit, as well as the reasons for the choice. For example, an LRP could consider the post-1998 literature only, because an LRP describing the literature up to that year already exists, or because the methods reviewed were first introduced in that year. In some cases, LRPs can be limited to specific contexts, for example, a country or category of countries, because the context may have an important impact on results. For example, the impact of land use on travel behaviour in the USA can differ from several EU countries because of differences in the public transport system, cycling culture, fuel prices, and the availability of sidewalks, legitimating an LRP on studies carried out in the USA or (a selection of) EU countries only.

Often snowballing is used, and this should be made explicit. Forward snowballing implies finding citations to a paper, whereas backward snowballing implies finding citations in a paper (Jalali & Wohlin, 2012). Even if the selection of papers to be included in the review is based on more subjective criteria, such as personal knowledge, extensive research in an area, brainstorming with experts, and other open-ended approaches, there is a methodological section that needs to be written on the process by which papers have been selected, together with comment and reflection on the strengths and weaknesses.

A search often results in too many papers being found for inclusion in the review, even after narrowing down the scope. We do not provide a precise threshold value, but LRPs in most cases might have a minimum threshold of 30 papers cited, and it is unlikely that more than 100 papers would be covered in the field of transport. If there are 'too' many papers, the solution may be to not include all papers, but to impose a (stratified) selection. This process should also be made explicit, and there should be a clear rationale to the logic of the process adopted for final selection. Reasons could relate to impact of papers (e.g. measured by citations — total or per year), geographical area, quality, whether the paper is recent or not, whether it is seminal or not, and many more criteria.

There are more methodological issues that deserve attention. As authors of LRPs should avoid criticising authors of original papers for things they did (not) do that do not match the scope of the LRP. For example, an LRP may review a paper on different levels of cycling, but an empirical study may only focus on utilitarian cycling, excluding recreational cycling. Excluding recreational cycling then should not be a criticism, but just an observation. The same applies to authors of empirical papers having used methods that were considered state of the art at the time of doing the research that in recent years have been considered as not the state of the art anymore. Reviewers should avoid simple averaging quantitative results. For example, if multiple studies review the impact of one variable on another, but some studies are based on only a few cases, whereas others include many more, averaging is misleading. It is better to present the individual results, combined with the number of cases, or weighing results in case of calculating averages. If averages are presented we also recommend presenting the range of results because probably the range is at least as important as the average. It is also very important to make explicit if conclusions and interpretations are provided by the authors of the original papers, or by the authors of the LRP.

Finally, a discussion on the ‘why’ behind results can be very helpful for the reader. This may be speculative to some extent, as long as this is made explicit. Speculation on reasons for patterns in the results may not only be helpful in trying to understand these patterns, but may also inspire readers in their own future research.

## 5. The Structure of a Paper

The LRP can be written in many different ways, and here we present a set of options but no recommended practice. The introduction can be very similar to a conventional research paper, discussing the background of the topic, what is already known in terms of the main lines of enquiry, the gap(s) in the literature, the motivation and aim of the paper related to the gap(s), the research questions, and for whom is the review targeted. The precise scope should be explained, preferably presented in a clear storyline. The methodology to be used could also be included in the introduction, but in light of the comments in Section 4 there may be a case for a separate methodology section. If there are already literature reviews in the same area this needs to be made explicit, as well as the position of the current LRP compared to those previously published LRPs in the same area.

Next, one would expect to see the presentation of an overview of the literature reviewed, often in the form of a table or a series of tables. Several ‘templates’ for such a presentation can be found. One common template for structuring the inputs to the review is to have as columns: author(s), descriptive characteristics such as the year of publication, geographical area, and sample size, but not the results of the review. The rows below are the papers/sources to be reviewed. For example, Hunt, Kriger, and Miller (2005) reviewing land-use transport frameworks used six columns to describe the software developed, the lead researcher, the history of the particular approach, the data platform, the commercial availability, and the support for the software. Six different models were selected, and they are then explicitly compared and commented on under each of the headings outlined above. Another template could be where the rows are *a priori* clusters of papers, for example, by world region or methodology. If a table is very long, it can be included as an appendix (see, for example, Salomon & Singer, 2014, where the authors review a range of travel measures over four time points).

In the case of papers with an empirical or methodological focus, the results of the review can also be presented in a table form, especially if the number of sources is relatively large, more than ten being a rough indication. This again can be structured with the papers/sources as rows. The columns provide the main content of the papers, and these are clearly linked to the aims of the review. For example, Nicolaisen and Driscoll (2014) use this approach to structure their ex post analysis of travel demand in a series of tables that systematically compare the results of the different studies by a consistent set of metrics.

The objective in using different formats and summary tables is to synthesise large amounts of information in a clear and concise way that makes it easy for the reader to both understand and to make comparisons between the broader approaches being reviewed. It is much easier to summarise with the use of tables, graphics, and other illustrative material, and it should also make the narrative easier to follow for the reader.

There are more options to structure the review paper than by source. It is also an option to have ‘results’ or content-related clustering as the guiding principle. For

example, if the papers review methods, it can also be that the first column presents key methods, and for each method then a description or typology is given, and next a column presents the sources that apply the method. Goodwin et al. (2004) in an LRP on price and income elasticities structure some of the content of their paper by distinguishing between the short- and the long-term elasticities. In case there are too many sources to be included in the LRP (see above), the author can decide to only present example papers, but explanation would be required.

Tables are not the only means to present substantial amounts of information, and in many cases, the use of text does the job just as well. Whether in table form or through other presentation devices, there are many options to structure the results section, depending on the aim of the paper. Examples would include the research area, the research period, the empirical focus, method(s), results, theories, etc. And in some cases figures can clearly present findings.

One observation is that authors of LRPs sometimes overlook the ‘obvious results’ that are often, but not necessarily, descriptive and characterise the body of literature in general terms. Examples of ‘obvious results’ might include the omission of studies published before a certain date originating from the USA, or that all studies found significant impacts of variable A on variable B, or that 80% of the studies were carried out after 2005, or that recent papers apply other methodologies. What might seem obvious to some needs to be explained, as it might be less obvious to others. A general tip is to present those obvious results in an early stage in the results section. A key element in the selection of papers is the audience for the LRP and their level of prior knowledge on the topic, as this determines the decision as to whether to include or exclude particular comments and results. This issue is difficult to give advice on, as the purpose of the LRP is to appeal to a wide readership that includes both experts in the subject area (for updating) and newcomers to the subject area (for more general background). The final decision here must reside with the author(s).

## 6. Final Remarks: Journal Choice, Abstract, Rejections of LRPs

Before writing any journal paper it is important to think about the journal of first choice for submission. Several transport journals do not (or only in very exceptional cases) accept LRPs, and examples here would be *Transportation Research Parts A and B*, *Transport Reviews* and *Transportation Research Part E: Logistics and Transportation Review* explicitly have ‘review’ in the journal title. Journals like the *Journal of Transport Geography* and *Urban Studies* also accept LRPs. Before choosing the journal of first choice it is wise to read recent examples of LRPs published in that journal, to get an impression of the tradition for LRPs. Some journals impose a maximum number of words for any paper and this limit needs to be checked, and all journals give authors guidance on the scope and expectations. Word limits for any paper often create problems, as authors have difficulties in keeping to externally imposed constraints, and an LRP is no exception. We have the impression it is even more difficult to stay within the word limit in case of an LRP, as there is always more to tell. Referees are also fond of asking for more material in an LRP, but are not so keen on suggesting what parts of a review paper should be shortened or omitted.

In most cases it is helpful that the abstract of an LRP not only describes the paper but also includes the main conclusions and added value. We realise that

different journals have different guidelines for the length of an abstract. In case of relatively long abstracts it could also include the background/introduction of the topic, its research questions, and its methodologies. This paper is intended to help aspiring writers of reviews to think about how writing an LRP can best be approached, and its purpose has been to highlight both the positives and the negatives in an open and informative way. The main reasons for rejection are not difficult to summarise. The key element is whether the paper is really a review or more of an overview, and here the key element is the added value. It should be written in an authoritative and constructively critical narrative. Another reason for rejection is that its aims are not met. The paper should clearly state its objectives in the introduction, and the conclusion should return to these objectives to assess whether they have been achieved. A third reason for rejection is that the paper has not been fully developed, as it is really an early draft. This means that the structure is weak, the evidence is partial, and there is no real content or thought in the LRP. In turn, this may result in a paper that is poorly constructed and has been submitted for external refereeing prematurely. The rejection levels are increasing for all major academic journals, and transport journals have rejection rates of between 50% and 90%. However, we see no reason why it should a priori be more difficult to get an LRP accepted in a high-quality transport journal than getting a conventional research paper accepted.

In summary, the LRP should primarily be a review paper that covers a wide range of literature from an authoritative and critical perspective, and that it comes to a set of conclusions that are supported by the evidence cited and adds value to the debate. Ideally, the paper should also be readable, interesting, and even exciting to read. Because of the high added value for readers, our intention is that this paper inspires potential authors to write an LRP that follows the *Transport Reviews* golden rule: a good review “takes a comprehensive overview of a subject, bringing together the material and coming to some interesting conclusions” (<http://www.tandf.com/journals/printview/?issn=0144-1647&subcategory=GE250000&linktype=1>).

## Acknowledgement

We thank three anonymous reviewers for their valuable comments on our draft paper.

## Disclosure statement

No potential conflict of interest was reported by the authors.

## References

- Avni, A., Burley, P., Casey, P., Cherney, J., Christiansen, L., Saunders Daly, J., ... Yu, H. (2015, March). *Literature searches and literature reviews for transportation research projects. How to search, where to search, and how to put it all together: Current practices*. Transportation Research Board, Transportation Research Circular Number E-C194. Retrieved April 7, 2015, from <http://www.trb.org/Publications/Blurbs/172271.aspx>
- Banister, D., Anderton, K., Bonilla, D., Givoni, M., & Schwanen, T. (2011). Transportation and the environment. *Annual Review of Environment and Resources*, 36, 247–270.
- Brons, M., Nijkamp, P., Pels, E., & Rietveld, P. (2008). A meta-analysis of the price elasticity of gasoline demand: A SUR approach. *Energy Economics*, 30(5), 2105–2122.

- Cao, X., Mokhtarian, P. L., & Handy, S. L. (2009). Examining the impacts of residential self-selection on travel behaviour: A focus on empirical findings. *Transport Reviews*, 29(3), 359–395.
- Denney, A. S., & Tewksbury, R. (2013). How to write a literature review. *Journal of Criminal Justice Education*, 24(2), 218–234.
- Dosi, G. (1982). Technological paradigms and technological trajectories: A suggested interpretation of the determinants and directions of social change. *Research Policy*, 11, 147–162.
- Freeman, C., & Louca, F. (2001). *As time goes by: From the industrial revolutions to the information revolution*. Oxford: Oxford University Press.
- Geels, F. W. (2011). The multilevel perspective on sustainability transitions: Responses to seven criticism. *Environmental Innovation and Societal Changes*, 1, 24–40.
- Geurs, K. T., & van Wee, B. (2004). Accessibility evaluation of land-use and transport strategies: Review and research directions. *Transport Geography*, 12(2), 127–140.
- Givoni, M. (2006). The development and impact of the modern high speed train. *Transport Reviews*, 26(5), 593–612.
- Goodwin, P., Dargay, J., & Hanly, M. (2004). Elasticities of road traffic and fuel consumption with respect to price and income: A review. *Transport Reviews*, 24(3), 275–292.
- Hunt, J. D., Kriger, D. S., & Miller, E. J. (2005). Current operational urban land-use-transport modelling frameworks: A review. *Transport Reviews*, 25(3), 329–376.
- Jalali, S., & Wohlin, C. (2012, September 19–20). *Systematic literature studies: Database searchers vs. backward snowballing*. Paper presented at the international conference on Empirical Software Engineering and Measurement, ESEM'12, Lund, Sweden. Retrieved June 10, 2015, from <http://www.wohlin.eu/esem12a.pdf>
- Mokhtarian, P. L., & Cao, X. (2008). Examining the impacts of residential self-selection on travel behavior: A focus on methodologies. *Transportation Research Part B*, 42(3), 204–228.
- Nelson, R. R., & Winter, S. G. (1982). *An evolutionary theory of economic change*. Cambridge, MA: Belnap Press.
- Nicolaisen, M. S., & Driscoll, P. A. (2014). Ex-post evaluations of demand forecast accuracy: A literature review. *Transport Reviews*, 34(4), 540–557.
- Nill, J., & Kemp, R. (2009). Evolutionary approaches for sustainable innovation policies: From niche to paradigm? *Research Policy*, 38, 668–680.
- Pucher, J., & Buehler, R. (2008). Making cycling irresistible: Lessons from the Netherlands. *Denmark and Germany: Transport Reviews*, 28(4), 495–528.
- Salomon, I., & Singer, M. E. (2014). Informal travel: A new conceptualization of travel patterns? *Transport Reviews*, 34(5), 645–662.
- Scheepers, C. E., Wendel-Vos, G. C. W., den Broeder, J. M., van Kempen, E. E. M. M., van Wesemael, P. J. V., & Schuit, A. J. (2014). Shifting from car to active transport: A systematic review of the effectiveness of interventions. *Transportation Research Part A*, 70, 264–280.
- Schwanen, T. (2013). Sociotechnical transition in the transport system. In M. Givoni & D. Banister (Eds.), *Moving towards low carbon mobility* (pp. 231–254). Cheltenham: Edward Elgar.
- Shove, E., & Walker, G. (2010). CAUTION! Transitions ahead: Politics, practice, and sustainable transition management. *Environment and Planning A*, 39, 763–770.
- Steward, B. (2004). Writing a literature review. *British Journal of Occupational Therapy*, 67(11), 495–500.
- Urry, J. (2011). The system of automobility, theory. *Culture and Society*, 21, 25–39.
- Van Acker, V., Van Wee, B., & Witlox, F. (2010). When transport geography meets social psychology: Toward a conceptual model of travel behaviour. *Transport Reviews*, 30(2), 219–240.
- Van Meeteren, M., Poorthuis, A., Derudder, B., & Witlox, F. (in press). Pacifying Babel's Tower: A scientometric analysis of polycentricity in urban research Urban Studies. *Urban Studies*.
- Webster, J., & Watson, R. T. (2002). Analyzing the past to prepare for the future: Writing a literature review. *MIS Quarterly*, 26(2), Xiii–Xxiii.
- Yang, H., & Bell, M. G. H. (1998). Models and algorithms for road network design: A review and some new developments. *Transport Reviews*, 18(4), 257–278.

# The Ordinal Nature of Emotions

Georgios N. Yannakakis

University of Malta

georgios.yannakakis@um.edu.mt

Roddy Cowie

Queen's University, Belfast

r.cowie@qub.ac.uk

Carlos Busso

The University of Texas at Dallas

busso@utdallas.edu

**Abstract**—Representing computationally everyday emotional states is a challenging task and, arguably, one of the most fundamental for affective computing. Standard practice in emotion annotation is to ask humans to assign an *absolute* value of intensity to each emotional behavior they observe. Psychological theories and evidence from multiple disciplines including neuroscience, economics and artificial intelligence, however, suggest that the task of assigning reference-based (*relative*) values to subjective notions is better aligned with the underlying representations than assigning absolute values. Evidence also shows that we use reference points, or else *anchors*, against which we evaluate values such as the emotional state of a stimulus; suggesting again that ordinal labels are a more suitable way to represent emotions. This paper draws together the theoretical reasons to favor relative over absolute labels for representing and annotating emotion, reviewing the literature across several disciplines. We go on to discuss good and bad practices of treating ordinal and other forms of annotation data, and make the case for preference learning methods as the appropriate approach for treating ordinal labels. We finally discuss the advantages of relative annotation with respect to both reliability and validity through a number of case studies in affective computing, and address common objections to the use of ordinal data. Overall, the thesis that emotions are by nature relative is supported by both theoretical arguments and evidence, and opens new horizons for the way emotions are viewed, represented and analyzed computationally.

**Keywords**—emotion annotation; ranks; ratings

## I. INTRODUCTION

Describing everyday emotional states is a complex and non-trivial task [1], and therefore so is labeling data for affective computing [2]. Several particular challenges affect labeling. People's impressions of emotions are a basic source, but the channels through which people can externalize them are frustratingly narrow. Standard practice in affective computing involves *absolute* annotation. Categorical labels (e.g., happiness and anger) or attribute descriptors (e.g., arousal and valence) try to locate the state of the subject at a point in a tree or a continuous space. Annotators are doing that when they observe an emotional behavior, and report the label or score that best reflects their perception. However, emotions are inherently structured. They are about something [3], which is why context is crucial to understanding [4]. They are also situated within narratives [5]. Trying to assign numbers to emotions is not simply a noisy task: there are a multitude of theoretical and practical reasons to doubt that subjective notions function as numbers in the first place [6].

In spite of theoretical doubts, it made sense to explore the use of absolute annotation for representing and annotating

emotion. However, the outcome underlines the doubts. This paper makes the case for engaging with a fundamentally different theoretical stance, where emotions are regarded as intrinsically *relative*. If so, their annotation and analysis should follow the ordinal path. To support this thesis we first review the theoretical arguments from psychology, and the empirical evidence coming from other disciplines, which favor relative conceptions of emotion (Section II). We then consider measurement issues, involving reliability and validity, which show the advantages of a relative annotation approach (Section III). In Section IV we describe the different annotation data types and the various (good, bad and ugly) ways these can be processed within affective computing. Then in Section V we present the preference learning paradigm for deriving affect models from ordinal data as demonstrated via a plethora of applications. Successful case studies showcasing the benefits of relative annotation for videos, speech and games are also presented (Section VI). The paper concludes with a discussion on the most common objections to the use of ordinal annotation (Section VII). Taken together, the arguments and evidence make a case for a paradigm shift in the way emotions are described computationally, annotated, and modeled.

## II. WHY RELATIVE OVER ABSOLUTE?

Multiple disciplines, including philosophy, psychology, economics, and artificial intelligence have studied the problem of measuring subjective variables, and taking decisions based on the results. In this section we survey the way various disciplines have developed ideas about relative perspectives and their importance for our decision making, as a way of dealing with subjective notions such as affect.

### A. Psychological Perspectives

A long standing thesis in psychology holds that while humans are efficient at discriminating among options [7], that is not matched by their ability to assign accurate absolute values for the intensity of what we perceive—and therefore, contrary to what we might imagine, does not depend on it. We are particularly bad at giving absolute values when the estimates involve subjective variables, such as the tension, frequency and loudness of sounds, the brightness of an image, or the arousal level of a video [7].

The theory related to that issue comes from various strands, and has a long history. As a way of structuring it, it makes sense to begin with themes that have early roots. *Adaptation level theory* was an approach to sensory experience that took

shape in the 1940s, and was drawn together by Helson in 1964 [8]. The central idea is that experience signals departure from a default level, the adaptation level, which is a weighted mean of previous stimuli. It is striking that an early paper by Russell, who became the best-known advocate of a dimensional account, stressed the relevance of adaptation level theory to emotion [9]. Similar points are made in diverse theories, including *relative judgment models* [10] suggesting that experience with stimuli gradually creates our internal context, and discussions of *anchors* [11], against which we rank any forthcoming stimulus or perceived experience. The accounts agree that our choice about an option is driven by our internal ordinal representation of that particular option within a sample of options; not by any absolute value of that option [12]. A particularly well-developed account, due Stewart et al. [12], extended theories of relative judgment to economic decision making. It models the subjective value involved in a decision as the outcome of a series of pairwise ordinal comparisons within a sample of attribute values drawn from memory; which, in turn, determine the final rank of the decision within the sample of options. The theory explicitly rejects appeals to underlying psycho-economic scales. Instead, it offers a Helsonian picture. Binary, ordinal comparisons to material held in memory provide the basis of subjective value. The value reflects its rank in a sample biased towards recent experience, but also longer-term information about the distribution of relevant attributes.

There is a straightforward application of the above theories to affect annotation. As with sensory dimensions, affective estimates are relative to an adaptation level. The implication is that it is very doubtful to treat equal ratings at different points in a rating session as if they meant the same thing. More fundamentally, Helson considers how experiences are related. He considers it obvious that there are **different kinds of pleasantness or unpleasantness**. The dimensional character comes from the fact that **instances that are of different kinds can still be ordered**. Ordering involves comparison with reference experiences. It is part of the adaptation level paradigm that a body of recent experiences act as immediate referents, but Helson also points a longer-term process, linked to conditioning, that establishes more enduring comparators. On that picture, numerical descriptions are a proxy for describing where, in a sequence built by comparisons, a particular experience lies.

The picture of emotions' relative nature fits most of the observations that are outlined in this paper. The argument is emphatically not that the evidence forces us to accept that kind of picture. It is that it provides a valuable perspective on the problems with reliability and disputes about practice in the collection and analysis of data within affective computing. It is not unreasonable to suspect that they reflect the need to rethink models of the way affect works at quite a deep level.

## B. Other Perspectives

The concepts underlying this paper have been explored in several other disciplines beyond psychology. The results and evidence are relevant for the study of emotions but are not

covered in detail due to space considerations. In **marketing research** values are traditionally measured with the use of rank-based questionnaires. According to Rokeach [13] societal or ethical values are acquired, internalized and organized in a hierarchical manner. The ranking approach naturally helps the respondent to discover, reveal and crystallize [13] his/her hierarchy of values in a self-reporting manner. The empirical evidence in that area is strong. For instance, a large scale study involving over 3,500 students across 19 counties [14] compared ratings and rankings for addressing the recurring problems of response style differences and language biases in cross-national research. The findings support the ranking approach: they show that it is more effective at reducing response biases in cross-cultural settings.

In **neuroscience**, Damasio [15] reports extensive experiments on the role of emotion in decision making. They imply that each time we are presented with a **stimulus**, we construct and store an anchor (or a *somatic marker*) which is eventually a mapping between the presented stimulus and our affective state. We then use these somatic markers as drivers for making choices between options. Given its unique role, affect can naturally guide our attention towards preferred options and, in turn, simplify the decision process for us. There is also evidence in monkeys [16] suggesting that their brain—in particular, the orbitofrontal cortex (OBC)—encodes values in a relative fashion. Similar results have been reported for the human medial OBC [17].

Arguments in **philosophy** also emphasize the fundamental role of comparison in subjectively defined values. The ***ceteris paribus*** (or *everything else being equal*) preference theory by Hansson [18] attempts to offer a unified formal structure of values and norms. Hansson claims that most of our daily decisions and choices are based on preferences *ceteris paribus* of two relata, A and B—as the building block of our decision making process. It rests on the problem of comparing structures with multiple attributes—which, as noted at the beginning of the paper, emotional experiences typically do, because they involve a context and a narrative.

The notion of **preference** is nowadays central in **artificial intelligence** and **machine learning** [19]. The theoretical grounding of learning from preferences [20] is based on humans' limited ability to express their preferences *directly* in terms of a specific value function [21]. That limitation holds even if the underlying scale of the notion we wish to assess is ordinal (e.g., in the case of ratings). This inability is mainly due to the **subjective nature of a preference** and the notion we express a preference about, and the substantial cognitive load required to give a specific value for each one of the available options we have to select from; and (recalling Hansson) each one of the options is characterized by a number of attributes (or the context) that we consider. Thus instead of rating our options directly it is far *easier* and more *natural* to express preferences about a number of **limited options**; and this is what we end up doing normally. As the **relative comparison between pairs of options** is less demanding (cognitively) than the **absolute assessment** of a set of single options, pairwise

preferences are easier to specify than exact value functions about the available options.

### III. PERFORMANCE MEASURES

This paper argues for the advantages of ranks as an emotion annotation tool. Clearly, that depends on identifying a measure of performance that shows these advantages. In this section we outline the two core criteria that against which annotation strategies can be measured: *reliability* and *validity*.

Inter-rater reliability is the degree of agreement among a number of annotators. This performance measure yields superior results for relative (rank-based) annotation, compared to other absolute annotation methods, as showcased by the case studies detailed in this paper. Testing whether the annotator is consistent over time (test-retest reliability) is also relevant for the thesis of this paper (see Section VI).

The validity of annotation is the degree to which the annotation construct measures the phenomenon we claim we measure. While interlinked to an extent, validity and reliability are different notions; the latter measures the degree to which our observations about a phenomenon are consistent. Validity in this paper is measured by the process of cross-validation in statistics and machine learning. Cross-validation examines the degree to which the result of a statistical analysis on data can generalize to unseen (independent) data. Several case studies in Section VI, and others in the literature showcase the superior generalizability of ordinal approaches to modeling affect.

### IV. DATA ANALYSIS FOR AFFECT MODELING

There are both theoretical and empirical arguments in favor of ordinal approaches to affect annotation and affect modeling. If so, obtaining ordinal labels in the first place would seem to be the ideal approach. That is not always possible, though. So, how should we process other data types, and how should we machine learn from data types other than ordinal? Following the taxonomy of Stevens [22] we can distinguish *three data types* that we can obtain from an emotion annotation task: *interval*, *nominal*, and *ordinal*. The next sections is what the thesis of the paper implies for the various data analysis practices followed in affective computing. That leads to an outline that covers the three different data types used, and considers the *good*, the *bad* and the so-called *ugly* data analysis practices associated with each. These practices are depicted in Fig. 1, respectively, as white, dark gray and light gray table cells.

#### A. Annotations are Interval

Interval data represent an affect state or dimension with a scalar value or a vector of values. Intervals are often confused with ratings and the terms are used interchangeably; however, ratings are not interval but rather ordinal values [6]. The most popular rating-based question is a Likert item [23] in which users are asked to specify their level of agreement with a given statement. Popular rating-based questionnaires for affect annotation include the Geneva Wheel model [24] and the Self-Assessment Manikin (SAM) [25]. When annotations come in

| Treating Column as Row | Interval                            | Nominal                                     | Ordinal                                |
|------------------------|-------------------------------------|---|--|
| Interval               | Ignores ordinal nature of emotion   | Impossible if no underlying order available | Introduces subjective reporting biases |
| Nominal                | Introduces split criterion biases   | Emotions are not necessarily discrete       | Introduces split criterion biases      |
| Ordinal                | Respects ordinal nature of emotions | Impossible if no underlying order available | Respects ordinal nature of emotions    |

Fig. 1. Practices in affective modeling: Treating column data types as row data types. White, dark gray, and light gray table cells, respectively, illustrate the *good*, *bad* and the *ugly* practices according to the thesis of this paper. By good we refer to approaches that are theoretically sound and compatible with the key message of the paper. By bad we refer to approaches that are technically flawed or even impossible and are also incompatible with the ordinal approach advocated in this paper. Finally, by ugly (perhaps too aggressively) we refer to approaches that are possible but nevertheless incompatible to the key message of the paper.

an interval form we can treat them as such or alternatively treat them as nominal or ordinal data.

If interval data is treated as such then a form of regression is naturally implied. For instance, one can think of attempting to approximate the absolute interval traces of arousal or valence using FeelTrace. This is a dominant practice in affective computing and it is also theoretically solid from a machine learning perspective. However, as advocated in this paper, the approach of approximating absolute values of subjective constructs such as emotions in models that misrepresent the ground truth of emotion.

Treating interval values as nominal data, instead, implies that one needs to first classify continuous annotations (e.g., from FeelTrace) and then create models via classification. This is another dominant practice in affect modeling (e.g., see studies on the SEMAINE dataset [26]) but recent evidence suggests that such practice introduces a multitude of biases in our data and thus takes us further away from the underlying ground truth [27]. Furthermore, creating dichotomized labels from interval data creates unavoidable problems where similar samples around the boundary are artificially placed in different classes [28].

Any attempt to derive an ordinal scale from interval data that characterize subjective notions appears to be a good practice to follow [29]–[31]. Several studies have transformed values of affect to ordered ranks and then derived affect models via preference learning: the transformation improves cross-validation capacities [27], [32], [33].

#### B. Annotations are Nominal

The second annotation data type one may obtain comes in *nominal* (or class) form. Nominal data are mutually exclusive labels which are not ordered, such as sex or unordered affective

states. Note, however, that nominal data sometimes take the form of a *preference* involving two or more options (for instance, they may indicate preference for the timbre of one sound in a list, or the warmth of one image in a set). There, an order of preference is implied—or is inherent—and underlies the observations. Binary nominal data that have a meaningful underlying order can also be viewed as borderline nominal. Examples include yes or no answers to questions such as *do you think this is a sad facial expression?* or *is the user in a high- or a low-arousal state?* In all such instances we argue that data can be safely treated as ordinal.

Deriving interval scores out of nominal values seems flawed or impossible unless there is an underlying order across the classes; e.g., an attempt is presented in [34]. The approach, however, leveraged on individual evaluations instead of consensus labels. The key idea was to create a probabilistic score per emotional category by considering the inter-evaluator agreement. The framework also considered relationships between emotional categories. For example, a sample receiving the label *excitement* increases its *happiness* score since these emotions are related. This is only possible if individual evaluations are available; otherwise, converting nominal values into interval scores is not feasible or appropriate.

Nominal data is ideal for multi class machine learning problems when emotional content is described in terms of categorical emotions. The common approach in affective computing is to ask multiple evaluators to select an emotional category after watching or listening to a stimulus. The individual evaluations are then often aggregated creating consensus labels. Forced-choice responses where an evaluator has to select an emotion out of a list create inaccurate descriptors, however. Depending on the options, the same stimulus can be annotated with different emotions [35]. Furthermore, nominal labels do not capture any within-class differences (i.e., different shades of happiness). As a result, the nominal labels tend to be noisy yielding poor inter-rater agreement, especially when the list of emotions is large [36].

An order cannot be easily derived from classes which are unordered—e.g., happiness and sadness. Indicatively, Lotfian and Busso [34] used a probabilistic score to define preferences between samples. The study established preferences when the difference between the probabilistic score of two samples was greater than a margin. On a similar basis, Cao et al. [37] also derived preferences from categorical emotions; in their study, every sentence labeled as happy was preferred over sentences labeled with another emotion. One drawback of this approach, however, is that it is not possible to establish preferences between samples from the same class. We argue for a more direct approach: to ask annotators to rank samples directly (e.g., is sample *A* happier than sample *B*?).

### C. Annotations are Ordinal

*Ordinal* data can be obtained via rank-based annotation protocols. The annotator is asked to rank a preference among options such as two or more images, musical pieces [31], sounds [38], or video screenshots [39], [40]. On its simplest

form, the annotator compares two options and specifies which one is preferred under a given statement (*pairwise preference*). With more than two options, the annotator is asked to provide a ranking of some or all the options. Examples of rank-based questions include: *was that level more engaging than this level?* *which facial expression looks happier?* *is the user more aroused now?*

Data obtained through the common rating-based annotation tools in affective computing such as FeelTrace or SAM is ordinal by nature [6]. Such data is generally treated as interval values, however—for instance, by averaging the obtained annotation values. While this is the dominant practice in psychometrics at large, there is extensive evidence for its invalidity and the numerous subjective reporting biases such analysis introduces to data [6], [29], [41].

Another popular practice is to **treat ordinal data as nominal and view the problem as a classification task**. Recent studies comparing the use of ordinal affect labels as ordinal against the use of ordinal labels as classes showcase the superiority of the first in yielding more general models of affect [27].

Finally treating ordinal data as ranks and viewing the problem of affect modeling as a preference learning task both respects the nature of the data and yields affect models of **supreme validity** [27], [33] and **reliability** [40]. The studies presented in Section VI provide additional evidence for the superior nature or relative affect annotation and its analysis for affect modeling.

## V. SO, I HAVE RANKS; HOW DO I DERIVE MY MODELS?

Given the different data types that we can obtain from our annotations, the next step is naturally to process it statistically or derive affect models which rely on this data. A popular objection to the use of ordinal labels is the lack of statistical tools and methods to process them. Section VII addresses common objections directly, but this section focuses on **preference learning (PL)** [19], [20], the natural approach to process ordinal (affect annotation) data and derive (affect) models from this data. **PL is a subfield of supervised learning dedicated to the processing of ordinal labels.** The PL paradigm as an approach for affective modeling was first introduced by Yannakakis back in 2009 [30]. Since then numerous studies in affective computing have used PL for affect detection and retrieval through images [42], [43], videos [39], [44], music [31], [45], sounds [38], speech [27], [37], [46], games [27], [47], [48] and text [49].

There are several algorithms and methods available for the task of preference learning. Most of them reduce the problem to pairwise comparisons where the task is to determine whether one sample, *A*, is preferred over another sample, *B*, (i.e.,  $A \succ B$ ). The results of the pairwise comparisons are used to rank the samples. It is important to note that *any* supervised learning algorithm can be converted to a PL problem by using an appropriate formulation. Linear statistical models, such as linear discriminant analysis and large margins, and non-linear approaches, such as Gaussian processes, shallow and deep

artificial neural networks, and support vector machines, are applicable for learning to predict ranks.

A popular derivation for PL consists of using binary classifiers. Let  $\phi$  be the feature vector of sample  $x$ . If  $x_i$  is preferred over  $x_j$  (i.e.,  $x_i \succ x_j$ ), the objective is to find a hyperplane  $w$  such that  $w(\phi_i - \phi_j) > 0$ , which is equivalent to a binary classification problem where the features are the subtraction of their respective feature vectors. This problem can be solved by any binary classifier; e.g., RankSVM is the equivalent PL method for support vector machines (SVMs) [50].

An alternative formulation for PL is training a function  $f$  that maintains a higher preference for the preferred option; for example, if  $x_i \succ x_j$  then  $f(\phi_i) > f(\phi_j)$ . There are several approaches to create this function: for example, it can take a parametric distribution as done with Gaussian processes [51], or can be learned from data using deep learning structures as performed via convolutional neural networks [47], [52] or via RankNet [53], or via neuroevolution [27], [30]. Studies have demonstrated that all aforementioned methods provide compelling results [30], [34], [46], [52], [54]. For the interested reader, a number of PL methods including RankSVM, neuroevolutionary preference learning and PL via backpropagation are contained in the preference learning toolbox (PLT) [55]. PLT is an open-access toolkit built and constantly updated for the purpose of easing the processing of ordinal labels.

#### A. Preference Learning for Affective Computing: Applications

Any application in emotion recognition can be formulated as a ranking problem in which PL algorithms are trained to predict ordinal labels. Examples of applications include forensic analysis where the goal is to prioritize the videos or audio to be analyzed by selecting a subset of recordings with target emotional content (e.g., threatening behaviors). Another example is in identifying emotionally salient regions, relying on relative emotional changes [56]. Computational tools that are able to rank emotions are also suitable for emotion retrieval, where the goal is to identify examples associated with a given emotional content [34]. Applications of emotional retrieval include solutions for health care domains [57], [58]. In longitudinal studies relying on remote assistant technologies, rank-based emotion retrieval can provide an ideal framework for healthcare practitioner to identify and review relevant events from patients with emotional disorders. Emotion retrieval from speech can facilitate better solutions for call centers. It can also facilitate the collection of natural emotional speech databases [59]. Emotion-aware recommendation systems are also an important application area for PL using ordinal labels (e.g., selecting music or sounds conveying emotions that match the current affective preference of the user [31], [38]).

The breadth of applications expand to video-based, [39], [40], speech-based [56] or physiology-based [60] emotion recognition for health, educational or entertaining [41] purposes. The next section covers a few successful applications

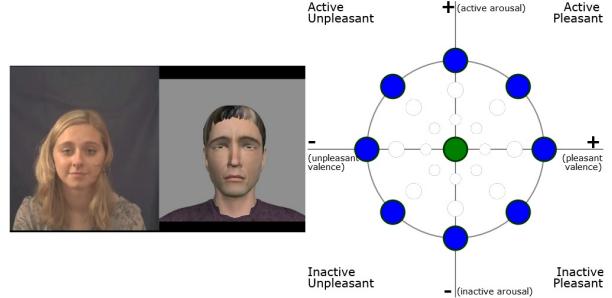


Fig. 2. *AffectRank*: the rank-based annotation tool introduced in [40]. *AffectRank* is inspired by *FeelTrace* but it allows the real-time annotation of arousal and/or valence in a relative fashion.

directly showcasing the benefits of ordinal annotation and processing for affect modeling.

## VI. CASE STUDIES RELEVANT FOR AFFECTIVE COMPUTING

In this section we outline a number of studies across various domains of affect computing that support the main argumentation and evidence of this paper. None of these studies is new; however, they are put together for the first time, thereby, collectively making our thesis stronger. We focus on important affect annotation studies that compare ordinal annotations against class-based and/or rating-based protocols within the domains of video, speech, and game experience annotation. Please note that the list is not exhaustive, because space is limited.

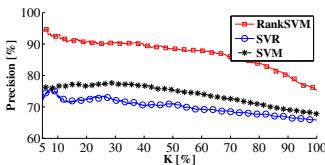
### A. Videos

While Metallinou and Narayanan [29] have long indicated the need of tools that would allow for a relative annotation of videos it is only very recently that such tools were introduced. The annotation tool named *AffectRank* [40] is a freely-available<sup>1</sup>, rank-based version of *FeelTrace* which asks the annotator to indicate a *change* in arousal and/or valence while watching a video. The evaluation study of [40] compared the inter-rater reliability between *FeelTrace* and *AffectRank* for the video annotation of two datasets: the SEMAINE [61] and the Eryi game dataset. The obtained results validate the hypothesis that *AffectRank* provides annotations that are significantly more reliable than the annotations obtained from *FeelTrace* (see Fig. 2). *AffectRank* yields superior reliability even when *FeelTrace* ratings are treated as ordinal data. The key findings of [40] further support the thesis of this paper by demonstrating that the dominant practice in continuous video affect annotation via rating-based labeling has negative effects.

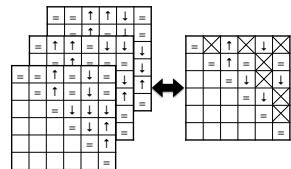
### B. Speech

Recent work in speech-based affect recognition has demonstrated the benefits of using PL with ordinal labels [32]–[34], [46]. Using time-continuous evaluations for arousal and valence provided by *FeelTrace*, the above studies defined

<sup>1</sup><https://github.com/TAPeri/AffectRank>



(a) Rankers versus other methods



(b) Finding Trends in Traces

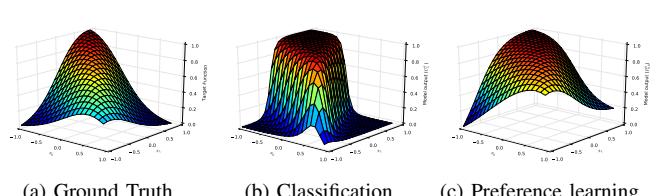
Fig. 3. Case studies on speech: (a) improved *precision at K* ( $P@K$ ) of RankSVM over regression (SVR) and binary classification (SVM) for arousal [32], (b) QA framework to identify trends from emotion annotation traces [33].

preferences between pairs of speech samples and compared PL (via RankSVM) against binary classification and regression for modeling arousal and valence. The task consisted of determining whether the value of the attribute of one sample was above or below the median value across the corpus (i.e., median split). This formulation applies directly to binary classifiers, where the positive and negative classes are defined according to the median split. For implementing regression, the predicted scores were sorted by selecting the samples at the top and bottom of the list. For preference learning, samples were ranked according to the emotion attributes, selecting samples in the extremes. The evaluation demonstrated that preference learning provided over 10% increase in cross-validation performance compared to the other two methods (see Fig. 3a). The evaluation also revealed two important observations. First, PL makes better use of the training set. Even when the margin that defines a preference is large, most of the data is still included in the ordinal dataset. Second, the results seem to saturate for RankSVM as the number of pairwise comparisons increases over 5,000 in the training set. We expect that deep architectures will be able to handle a bigger dataset, achieving better results [27], [46], [47].

In another study, it was proposed to define ordinal labels by considering trends in the time-continuous labels [33]. Each dialog is annotated by multiple evaluators creating a trace per rater. A common observation is that these traces are noisy with low inter-evaluator agreement. Instead of averaging the traces across evaluators, the qualitative agreement (QA) framework [62] was used to identify segments where most of the evaluators agreed on trends (e.g., increase or decrease in the values of the traces). This framework leverages consistent information provided in the, otherwise, noisy traces. The emotion annotation traces are segmented into bins, and their average are compared creating an individual matrix per evaluator (right side of Fig. 3b). The arrows denotes increasing or decreasing trends between bins. All the individual matrices are then combined creating a consensus matrix with the consistent trends (left side of Fig. 3b). The core findings suggest that extracting ordinal labels with QA provides better classifiers, increasing the accuracy of the emotion rankers.

### C. Games

The literature on the benefits of ordinal annotation in video games is rich. Several studies have explored both first-person



(a) Ground Truth      (b) Classification      (c) Preference learning

Fig. 4. A hypothesized (artificial) ground truth function ( $z$ -axis) which is dependent on two attributes,  $x_1$  and  $x_2$  (Fig. 4a), the best classification model (Fig. 4b) and the best preference learned model (Fig. 4c) [27].

and third-person ordinal annotation of playing experience and player affect. Indicatively, ordinal annotation protocols have been explored in racing games [48], prey-predator [6], [27], [40], horror [38], and physical interactive games [41] among many other game genres. Most notably for the purposes of this paper, Yannakakis and Hallam compared rating versus ranking annotations of first person experience in both a prey predator and a physical interactive game [41]. Their subjects were asked to use 5-point (prey-predator) or 10-point (physical interactive game) Likert items versus a ranking protocol to answer questions about the experience of the games they just played. The affective states they explored spanned from fun to frustration, to excitement and boredom. Their key findings reveal that rater consistency (reliability) is higher when ranking protocols are used across both games. Further their evidence suggests that the order of answering affects ratings more than ranks. In other words, ranks yield higher degrees of test-retest reliability.

In another study, Martinez et al. [27] worked on the hypothesis that the best way of analyzing ratings of affect is to treat them naturally as ordinal data. To test their hypothesis they compared models of affect that are the result of converting ratings to classes (classification) versus ordinal models that are trained directly via preference learning. They used three datasets for their analysis: an artificial dataset, a dataset from the MazeBall game containing physiological signals and gameplay data [52] and the SAL [63] corpus which contained 739 1-second-long speech segments. The main findings of their study validate their hypothesis and further support the thesis of this paper. Models trained via preference learning outperform the classification models of affect in terms of cross-validation. Figure 4 showcases how much closer a preference learned model can reach a hypothesized (artificial) ground truth, compared to a classification model.

Importantly for the thesis of this paper, Holmgård et al. [60] compare different types of stress annotation with the aim of finding the best possible approximation to the underlying ground truth. In particular they compare annotations indicating the most stressful event in a game (class-based annotation) versus a rank-based approach by which subjects compare stress across game events. Their findings reveal that the ordinal annotations are more accurate predictors of the phasic driver of skin conductance which is assumed to be a reliable indicator of underlying stress.

## VII. CAN LESS BE MORE?

In this section we discuss a number of traditional objections to the use or ranks or relative constructs in affective computing and human computer interaction at large. There are certainly more objections we could discuss; however, we put an emphasis on the most important ones given the limited space.

**More is Better:** It is natural to believe that more information about a subjectively defined notion such as an emotion is a good property. It is also safe to believe that if one merely compares emotions in a relative fashion and does not specify an absolute value about them the resulting analysis suffers from lack of resolution and intensity. However, all studies presented in this paper, the evidence collected from other disciplines and, finally, the psychological framework about the relative nature of emotions collectively suggest the exact opposite: that *less is more*. It appears that the additional information offered by absolute annotation is only biasing the search for valid and reliable models of affect. Further, at first sight it might seem that the *intensity* of an emotion is completely lost if it is expressed in a relative fashion as it is only compared to an anchor or a reference point. Empirical evidence from this paper showcase that the intensity is not lost; it is instead lying under the provided ranks. The function that models affective ranks (e.g., a preference learned neural network [27], [47]) can directly output intensity values of the modeled affect. In other words, intensity is not only present when ordinal labels are used but is also free of reporting biases caused through absolute annotation.

**Anchors:** To rank means inherently to compare. To be able to compare one needs a point of reference. Ranking-based annotations require at least one reference point which is usually found as an option in the question. While the requirement of a reference point might seem a core limitation of rankings we argue that it is their biggest strength. As we discussed thoroughly in this paper it is theoretically grounded that humans maintain a baseline when using any type of reporting scheme. Be it a class or a rating question an annotator will make a comparison based on other items within the scale or earlier responses in similar questions and contexts. The power of ranks is that this baseline extraction process does not have to happen unconsciously or intuitively; it is forced. Further the baseline is not some mere approximation of preference indicated by a scale or distorted by memory; it is a real option one uses as a reference during the annotation. Once again, this property of ranks appears to be a limitation but it instead encapsulates one of their core strengths.

**Statistical Analysis:** Given their ordinal nature it is not always possible to apply conventional statistical methods to ordinal data. Standard descriptive statistics such as mean values and standard deviations are not applicable. Parametric tests are not applicable either. There are still, however, multiple data visualization methods and data processing techniques that span from classical correlation analysis to statistical tests for significance and further to modern machine learning approaches that are available for handling preferences and ranks.

This paper already covered several of the methods (Section V) but the reader is also referred to [6], [40] for more details.

## VIII. CONCLUSIONS

This paper has presented and supported the thesis that emotions are by nature *relative*. We do not claim that it is a novel thesis. On the contrary, we have taken pains to show that it reflects established ideas in many literatures—psychology, philosophy, neuroscience, marketing research, artificial intelligence and, not least, affective computing. The research in affective computing allows us to identify good and bad practices for the analysis of interval, nominal and ordinal annotations; and it provides several practical ways of processing ordinal affective labels via preference learning. It also provides studies, across various domains of affective computing, which showcase the advantages of treating emotions as relative notions. Our fundamental aim is to make it clear that this is not a fringe issue. Attempts to work with absolute annotation, including our own, have shown that problems we knew in principle do not turn out to be unimportant in practice. The cumulation of evidence says that it makes sense to look in a concerted way at the alternative that various teams, including our own, have been exploring.

## REFERENCES

- [1] R. Cowie, “Describing the forms of emotional colouring that pervade everyday life,” in *The Oxford Handbook of Philosophy of Emotion*, P. Goldie, Ed. Oxford, UK: Oxford University Press, January 2010, pp. 63–94.
- [2] R. Cowie, C. Cox, J. C. Martin, A. Batliner, D. Heylen, and K. Kar pouzis, “Issues in data labelling,” in *Emotion-Oriented Systems*, P. Petta, C. Pelachaud, and R. Cowie, Eds. Springer Berlin Heidelberg, July 2011, pp. 213–241.
- [3] P. Goldie, “Emotions, feelings and intentionality,” *Phenomenology and the Cognitive Sciences*, vol. 1, no. 3, pp. 235–254, September 2002.
- [4] L. Barrett, B. Mesquita, and M. Gendron, “Context in emotion perception,” *Current Directions in Psychological Science*, vol. 20, no. 5, pp. 286–290, October 2011.
- [5] P. Goldie, *The mess inside: narrative, emotion, and the mind*. Oxford, UK: Oxford University Press, September 2012.
- [6] G. N. Yannakakis and H. P. Martínez, “Ratings are overrated!” *Frontiers in ICT*, vol. 2, p. 13, 2015.
- [7] G. A. Miller, “The magical number seven, plus or minus two: some limits on our capacity for processing information.” *Psychological review*, vol. 63, no. 2, p. 81, 1956.
- [8] H. Helson, “Adaptation-level theory,” 1964.
- [9] J. A. Russell and U. F. Lanius, “Adaptation level and the affective appraisal of environments,” *Journal of Environmental Psychology*, vol. 4, no. 2, pp. 119–135, 1984.
- [10] N. Stewart, G. D. Brown, and N. Chater, “Absolute identification by relative judgment.” *Psychological review*, vol. 112, no. 4, p. 881, 2005.
- [11] B. Seymour and S. M. McClure, “Anchors, scales and the relative coding of value in the brain,” *Current opinion in neurobiology*, vol. 18, no. 2, pp. 173–178, 2008.
- [12] N. Stewart, N. Chater, and G. D. Brown, “Decision by sampling,” *Cognitive psychology*, vol. 53, no. 1, pp. 1–26, 2006.
- [13] M. Rokeach, *The nature of human values*. Free press, 1973.
- [14] T. Johnson, P. Kulesa, Y. I. Cho, and S. Shavitt, “The relation between culture and response styles: Evidence from 19 countries,” *Journal of Cross-cultural psychology*, vol. 36, no. 2, pp. 264–277, 2005.
- [15] A. R. Damasio, “Descartes’ error: Emotion, rationality and the human brain,” 1994.
- [16] L. Tremblay and W. Schultz, “Relative reward preference in primate orbitofrontal cortex,” *Nature*, vol. 398, no. 6729, pp. 704–708, 1999.
- [17] R. Elliott, Z. Agnew, and J. Deakin, “Medial orbitofrontal cortex codes relative rather than absolute value of financial rewards in humans,” *European Journal of Neuroscience*, vol. 27, no. 9, pp. 2213–2218, 2008.

- [18] S. O. Hansson, "What is ceteris paribus preference?" *Journal of Philosophical Logic*, vol. 25, no. 3, pp. 307–332, 1996.
- [19] S. Kaci, *Working with preferences: Less is more*. Springer Science & Business Media, 2011.
- [20] J. Fürnkranz and E. Hüllermeier, *Preference learning: An introduction*. Springer, 2010.
- [21] C. Domshlak, E. Hüllermeier, S. Kaci, and H. Prade, "Preferences in AI: An overview," *Artificial Intelligence*, vol. 175, no. 7-8, pp. 1037–1052, 2011.
- [22] S. S. Stevens, "On the Theory of Scales of Measurement," *Science*, vol. 103, no. 2684, pp. 677–680, 1946.
- [23] R. Likert, "A technique for the measurement of attitudes." *Archives of psychology*, 1932.
- [24] K. Scherer, "What are emotions? and how can they be measured?" *Social science information*, vol. 44, no. 4, pp. 695–729, 2005.
- [25] J. Morris, "Observations: Sam: The self-assessment manikin—an efficient cross-cultural measurement of emotional response," *Journal of advertising research*, vol. 35, no. 6, pp. 63–68, 1995.
- [26] S. Asteriadis, P. Tzouveli, K. Karpouzis, and S. Kollias, "Non-verbal feedback on user interest based on gaze direction and head pose," in *Proc. of Int. Workshop on Semantic Media Adaptation and Personalization (SMAP)*. IEEE Computer Society, 2007, pp. 171–178.
- [27] H. Martinez, G. Yannakakis, and J. Hallam, "Don't classify ratings of affect; rank them!" *Affective Computing, IEEE Transactions on*, vol. 5, no. 3, pp. 314–326, 2014.
- [28] S. Mariooryad and C. Busso, "The cost of dichotomizing continuous labels for binary classification problems: Deriving a Bayesian-optimal classifier," *IEEE Transactions on Affective Computing*, vol. 8, no. 1, pp. 119–130, January–March 2017.
- [29] A. Metallinou and S. Narayanan, "Annotation and processing of continuous emotional attributes: Challenges and opportunities," in *Automatic Face and Gesture Recognition (FG), 2013 10th IEEE International Conference and Workshops on*. IEEE, 2013, pp. 1–8.
- [30] G. N. Yannakakis, "Preference learning for affective modeling," in *Affective Computing and Intelligent Interaction and Workshops, 2009. ACII 2009. 3rd International Conference on*. IEEE, 2009, pp. 1–6.
- [31] Y.-H. Yang and H. H. Chen, "Ranking-based emotion recognition for music organization and retrieval," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 762–774, 2011.
- [32] R. Lotfian and C. Busso, "Practical considerations on the use of preference learning for ranking emotional speech," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2016)*, Shanghai, China, March 2016, pp. 5205–5209.
- [33] S. Parthasarathy, R. Cowie, and C. Busso, "Using agreement on direction of change to build rank-based emotion classifiers," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 11, pp. 2108–2121, November 2016.
- [34] R. Lotfian and C. Busso, "Retrieving categorical emotions using a probabilistic framework to define preference learning samples," in *Interspeech 2016*, San Francisco, CA, USA, September 2016, pp. 490–494.
- [35] J. A. Russell, "Forced-choice response format in the study of facial expression," *Motivation and Emotion*, vol. 17, no. 1, pp. 41–51, 1993.
- [36] C. Busso, M. Bulut, and S. Narayanan, "Toward effective automatic recognition systems of emotion in speech," in *Social emotions in nature and artifact: emotions in human and human-computer interaction*, J. Gratch and S. Marsella, Eds. New York, NY, USA: Oxford University Press, November 2013, pp. 110–127.
- [37] H. Cao, R. Verma, and A. Nenkova, "Speaker-sensitive emotion recognition via ranking: Studies on acted and spontaneous speech," *Computer Speech & Language*, vol. 29, no. 1, pp. 186–202, January 2015.
- [38] P. Lopes, A. Liapis, and G. N. Yannakakis, "Modelling affect for horror soundscapes," *IEEE Transactions on Affective Computing*, 2017.
- [39] Y. Baveye, E. Dellandrea, C. Chamaret, and L. Chen, "From crowd-sourced rankings to affective ratings," in *Multimedia and Expo Workshops, 2014 IEEE International Conference on*. IEEE, 2014, pp. 1–6.
- [40] G. N. Yannakakis and H. P. Martinez, "Grounding truth via ordinal annotation," in *Affective Computing and Intelligent Interaction (ACII), 2015 International Conference on*. IEEE, 2015, pp. 574–580.
- [41] G. Yannakakis and J. Hallam, "Rating vs. preference: a comparative study of self-reporting," *Affective Computing and Intelligent Interaction*, pp. 437–446, 2011.
- [42] W. Wang and Q. He, "A survey on emotional semantic image retrieval," in *IEEE International Conference on Image Processing (ICIP 2008)*, San Diego, CA, USA, October 2008, pp. 117–120.
- [43] S. Schmidt and W. G. Stock, "Collective indexing of emotions in images. A study in emotional information retrieval," *J. of the American Soc. for Information Science and Technology*, vol. 60, no. 5, pp. 863–876, 2009.
- [44] J. Kierkels, M. Soleymani, and T. Pun, "Queries and tags in affect-based multimedia retrieval," in *IEEE International Conference on Multimedia and Expo*, Amsterdam, The Netherlands, 2009, pp. 1436–1439.
- [45] K. Trohidis, G. Tsoumacas, G. Kalliris, and I. P. Vlahavas, "Multi-label classification of music into emotions," in *International Conference on Music Information Retrieval (ISMIR 2008)*, Philadelphia, PA, USA, September 2008, pp. 325–330.
- [46] S. Parthasarathy, R. Lotfian, and C. Busso, "Ranking emotional attributes with deep neural networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2017)*, New Orleans, LA, USA, March 2017, pp. 4995–4999.
- [47] H. P. Martínez and G. N. Yannakakis, "Deep multimodal fusion: Combining discrete events and continuous signals," in *Proceedings of the 16th Int. Conf. on Multimodal Interaction*. ACM, 2014, pp. 34–41.
- [48] S. Tognetti, M. Garbarino, A. Bonarini, and M. Matteucci, "Modeling enjoyment preference from physiological responses in a car racing game," in *Computational Intelligence and Games (CIG), 2010 IEEE Symposium on*. IEEE, pp. 321–328.
- [49] B. Pang and L. Lee, "Opinion mining and sentiment analysis," *Foundations and trends in information retrieval*, vol. 2, no. 1-2, pp. 1–135, 2008.
- [50] R. Herbrich, T. Graepel, and K. Obermayer, "Support vector learning for ordinal regression," in *International Conference on Artificial Neural Networks (ICANN 1999)*, Edinburgh, UK, September 1999, pp. 97–102.
- [51] W. Chu and Z. Ghahramani, "Preference learning with Gaussian processes," in *International conference on machine learning (ICML 2005)*, Bonn, Germany, August 2005, pp. 137–144.
- [52] H. P. Martínez, Y. Bengio, and G. N. Yannakakis, "Learning deep physiological models of affect," *Computational Intelligence Magazine, IEEE*, vol. 8, no. 2, pp. 20–33, 2013.
- [53] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender, "Learning to rank using gradient descent," in *Int. Conf. on Machine learning*, Bonn, Germany, 2005, pp. 89–96.
- [54] G. Yannakakis, M. Maragoudakis, and J. Hallam, "Preference learning for cognitive modeling: a case study on entertainment preferences," *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, vol. 39, no. 6, pp. 1165–1175, 2009.
- [55] V. E. Farrugia, H. P. Martínez, and G. N. Yannakakis, "The preference learning toolbox," *arXiv preprint arXiv:1506.01709*, 2015.
- [56] S. Parthasarathy and C. Busso, "Defining emotionally salient regions using qualitative agreement method," in *Interspeech 2016*, San Francisco, CA, USA, September 2016, pp. 3598–3602.
- [57] M. Kranzfelder, A. Schneider, S. Gillen, and H. Feussner, "New technologies for information retrieval to achieve situational awareness and higher patient safety in the surgical operating room: the MRI institutional approach and review of the literature," *Surgical Endoscopy*, vol. 25, no. 3, pp. 696–705, March 2011.
- [58] K. Pollak, R. Arnold, A. Jeffreys, S. Alexander, M. Olsen, A. Abernethy, C. Sugg Skinner, K. Rodriguez, and J. Tulsky, "Oncologist communication about emotion during visits with patients with advanced cancer," *Journal of Clinical Oncology*, vol. 25, no. 36, pp. 5748–5752, 2007.
- [59] S. Mariooryad, R. Lotfian, and C. Busso, "Building a naturalistic emotional speech corpus by retrieving expressive behaviors from existing speech corpora," in *Interspeech 2014*, Singapore, September 2014, pp. 238–242.
- [60] C. Holmgård, G. N. Yannakakis, H. P. Martinez, and K.-I. Karstoft, "To rank or to classify? annotating stress for reliable ptsd profiling," in *Affective Computing and Intelligent Interaction (ACII), 2015 International Conference on*. IEEE, 2015, pp. 719–725.
- [61] G. McKeown, M. Valstar, R. Cowie, M. Pantic, and M. Schroder, "The semaine database: Annotated multimodal records of emotionally colored conversations between a person and a limited agent," *Affective Computing, IEEE Transactions on*, vol. 3, no. 1, pp. 5–17, 2012.
- [62] R. Cowie and G. McKeown, "Statistical analysis of data from initial labelled database and recommendations for an economical coding scheme," September 2010, SEMAINE Report D6b.
- [63] K. Karpouzis, G. Caridakis, R. Cowie, and E. Douglas-Cowie, "Induction, recording and recognition of natural emotions from facial expressions and speech prosody," *Journal on Multimodal User Interfaces*, vol. 7, no. 3, pp. 195–206, 2013.

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/226120628>

# Capturing Player Enjoyment in Computer Games

Chapter · June 2007

DOI: 10.1007/978-3-540-72705-7\_8

---

CITATIONS

13

READS

131

2 authors, including:



Georgios Yannakakis

University of Malta

220 PUBLICATIONS 5,360 CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:



COMnPLAY-Science [View project](#)



C2Learn [View project](#)

# 1. Capturing Player Enjoyment in Computer Games

Georgios N. Yannakakis and John Hallam

Maersk Mc-Kinney Moller Institute  
University of Southern Denmark  
Campusvej 55, Odense M, DK-5230  
[{georgios, john}@mip.sdu.dk](mailto:{georgios, john}@mip.sdu.dk)

The current state-of-the-art in intelligent game design using Artificial Intelligence (AI) techniques is mainly focused on generating human-like and intelligent characters. Even though complex opponent behaviors emerge through various machine learning techniques, there is generally no further analysis of whether these behaviors contribute to the satisfaction of the player. The implicit hypothesis motivating this research is that intelligent opponent behaviors enable the player to gain more satisfaction from the game. This hypothesis may well be true; however, since no notion of entertainment or enjoyment is explicitly defined, there is therefore no evidence that a specific opponent behavior generates enjoyable games.

This chapter introduces a discussion of quantitative entertainment capture in real-time and presents two dissimilar approaches for modeling player satisfaction. Successfully capturing the level of entertainment during play provides insights for designing the appropriate AI methodology for entertainment augmentation in real-time. For this purpose, adaptive on-line learning methodologies are proposed and their strengths and limitations are discussed.

## 1.1 Introduction

Cognitive modeling within human-computer interactive systems is a prominent area of research. Computer games, as examples of such systems, provide an ideal environment for research in artificial intelligence (AI), because they are based on simulations of highly complex and dynamic multi-agent worlds [1.1, 1.2, 1.3]. Moreover, computer games offer a promising ground for cognitive modeling since they embed rich forms of interactivity between humans and non-player characters (NPCs) [1.4]. Being able to capture quantitatively the level of user (gamer) engagement or satisfaction in real-time can grant insights to the appropriate AI methodology for enhancing the quality of playing experience [1.5] and furthermore be used to adjust digital entertainment environments according to individual user preferences.

Motivated by the lack of quantitative models of entertainment, endeavors to measure and augment player satisfaction in real-time are presented in this chapter. More specifically, the open question of modeling entertainment during game play is discussed and the strengths and weaknesses of

previous attempts in the field are outlined. Herein entertainment is defined qualitatively primarily as the level of satisfaction generated by the real-time player-game opponent interaction — by ‘opponent’ we define any controllable interactive feature of the game. We view a game primarily as a learning process, and the level of entertainment is kept high when game opponents enable new learning patterns (‘not too easy a game’) for the player that can be perceived and learned by the player (‘not too difficult a game’) [1.6, 1.7]. On the same basis, according to [1.8] — within the axis of emotions varying from boredom to fascination — learning is highly correlated with interest, curiosity and intrigue perceived. The collection of these emotions is referred to as entertainment (or “fun”) in this chapter.

Two different approaches for quantitatively capturing and enhancing the real-time entertainment value of computer games are presented in this chapter: one based on empirical design of entertainment metrics and one where quantitative entertainment estimates are extracted using machine learning techniques, grounded in psychological studies. The predator/prey game genre is used for the experiments presented here; though it is argued that the proposed techniques can be applied more generally, to other game genres.

In the first, “empirical,” approach, a quantitative metric of the ‘interestingness’ of opponent behaviors is designed based on qualitative considerations of what is enjoyable in predator/prey games. A mathematical formulation of those considerations, based upon data observable during game play, is derived. This metric is validated successfully against the human notion of entertainment.

In the second approach, entertainment modeling is pursued by following the theoretical principles of Malone’s [1.9] intrinsic qualitative factors for engaging game play, namely *challenge* (i.e. ‘provide a goal whose attainment is uncertain’), *curiosity* (i.e. ‘what will happen next in the game?’) and *fantasy* (i.e. ‘show or evoke images of physical objects or social situations not actually present’) and driven by the basic concepts of the Theory of Flow [1.10] (‘flow is the mental state in which players are so involved in the game that nothing else matters’). Quantitative measures for challenge and curiosity are inspired by the “empirical” approach to entertainment metrics. They are represented by measures computed from appropriate game features based on the interaction of player and opponent behavior. A mapping between the quantitative values of these challenge and curiosity measures and the human notion of entertainment is then constructed using evolving neural networks (NNs).

The chapter concludes with a discussion of several remaining open questions regarding entertainment modeling and proposes future directions to answer these questions. The limitations of the presented methodology, and the extensibility of the proposed approaches of entertainment capture and augmentation to other genres of digital entertainment, are also discussed.

## 1.2 Capturing Entertainment

There have been several psychological studies to identify what is “fun” in a game and what engages people playing computer games. Theoretical approaches include Malone’s principles of intrinsic qualitative factors for engaging game play [1.9], namely challenge, curiosity and fantasy as well as the well-known concepts of the theory of flow [1.10] incorporated in computer games as a model for evaluating player enjoyment, namely *GameFlow* [1.11].

A comprehensive review of the literature on qualitative approaches for modeling player enjoyment demonstrates a tendency to overlap with Malone’s and Csikszentmihalyi’s foundational concepts. Many of these approaches are based on Lazzaro’s “fun” clustering which uses four entertainment factors based on facial expressions and data obtained from game surveys of players [1.12]. According to Lazzaro, the four components of entertainment are: hard fun (related to the challenge factor of Malone), easy fun (related to the curiosity factor of Malone), altered states (i.e. ‘the way in which perception, behavior, and thought combine in a collective context to produce emotions and other internal sensations’ — closely related to Malone’s fantasy factor) — and socialization (the people factor). Koster’s [1.7] theory of fun, which is primarily inspired by Lazzaro’s four factors, defines “fun” as the act of mastering the game mentally. An alternative approach to fun measure is presented in [1.13] where fun is composed of three dimensions: durability, engagement and expectations. Questionnaire tools and methodologies are proposed in order to empirically capture the level of fun for evaluating the usability of novel interfaces with children.

Kline and Arlidge [1.14] support Lazzaro’s findings since their studies on on-line massive multi-player games (e.g. Counter-Strike) identify the socialization factor as an additional component to Malone’s factors for “fun” game play experiences. Their clustering of player styles (‘player archetypes’) corresponds to these four dimensions of entertainment: *Warriors* are those who prioritize combat features and realism (closely related to Malone’s fantasy factor), *Narrators* are those who place priority on the plot, characters, and exploration but they do not like games that are challenging (closely related to Malone’s curiosity factor), *Strategists* are those that prioritize complex strategies, challenging game play and mastery (closely related to Malone’s challenge factor) and *Interactors* for whom competition and cooperation with other players is of the highest importance (socialization factor).

On that basis, some endeavors towards the criteria that collectively make simple on-line games appealing are discussed in [1.15]. The human survey-based outcome of that work presents challenge, diversity and unpredictability as primary criteria for enjoyable opponent behaviors.

Vorderer et al. [1.4] present a quantitative analysis (through an extensive human player survey on the Tomb Raider game) of the impact of competition (i.e. challenge) on entertainment and identify challenge as the most important determinant of the enjoyment perceived by video game players.

They claim that a successful completion of a task generates sympathetic arousal, especially when the challenge of the task matches the player's abilities. In their analysis, social competition (just like Lazzaro's people factor and Kline's and Arlidge's *Interactors*) appears to enhance entertainment. Their survey, however, is based on single game-task enjoyment evaluations rather than comparative evaluations of several scenarios.

The study by Choi et al. [1.16] ranks perceptual and cognitive fun as the top-level factors for designing "fun" computer games. For the former, it is the game interface that affects the player's perception (vividness and imagination) — this corresponds to Malone's fantasy factor. For the latter, it is the game mechanics (level of interactivity) that affect the player's cognitive process — which comprises the challenge and satisfaction factors. According to Choi et al. (*ibid.*), challenge and satisfaction appear as independent processes, in contrast to the views of Malone [1.9] and Yannakakis et al. [1.17] where satisfaction derives from the appropriate level of challenge and other game components. Moreover, in Choi et al.'s study, vividness and imagination (perceptual fun) appear more important entertainment factors for players of strategic simulation games and challenge and satisfaction (cognitive fun) appear more important for role-playing games.

As previously mentioned, research in the field of game AI is mainly focused on generating human-like (believable) and intelligent (see [1.3, 1.18] among others) characters. Complex NPC behaviors can emerge through various AI techniques; however, there is no further analysis of whether these behaviors have a positive impact to the satisfaction of the player during play. According to Taatgen et al. [1.19], believability of computer game opponents, which are generated through cognitive models, is strongly correlated with enjoyable games. Such implicit research hypotheses may well be true; however, there is little evidence that specific NPCs generate enjoyable games unless a notion of interest or enjoyment is explicitly defined.

Iida's work on metrics of entertainment in board games was the first attempt in the area of quantitative "fun" modeling. He introduced a general metric of entertainment for variants of chess games depending on average game length and possible moves [1.20]. Other work in the field of quantitative entertainment capture is based on the hypothesis that the player-opponent interaction — rather than the audiovisual features, the context or the genre of the game — is the property that contributes the majority of the quality features of entertainment in a computer game [1.6]. Based on this fundamental assumption, a metric for measuring the real time entertainment value of predator/prey games was designed, and established as efficient and reliable by validation against human judgement [1.21, 1.22]. Further studies by Yannakakis and Hallam [1.23] have shown that Artificial Neural Networks (ANN) and fuzzy neural networks can extract a better estimator of player satisfaction than a custom-designed (or designer-driven) one, given appropriate estimators of the challenge and curiosity of the game and data on human players'

preferences. Similar work in adjusting a game's difficulty include endeavors through reinforcement learning [1.24], genetic algorithms [1.25], probabilistic models [1.26] and dynamic scripting [1.27]. However, the aforementioned attempts are based on the assumption that challenge is the only factor that contributes to enjoyable gaming experiences while results reported have not been cross-verified by human players.

A step further to entertainment capture is towards games of richer human-computer interaction and affect recognizers which are able to identify correlations between physiological signals and the human notion of entertainment. Experiments by Yannakakis et al. [1.28] have already shown a significant correlation of average heart rate with children's perceived entertainment in action games played in interactive physical playgrounds. Moreover, Rani et al. [1.29] propose a methodology for detecting anxiety level of the player and appropriately adjusting the level of challenge in the game of 'Pong' based on recorded physiological signals in real-time and subject's self-reports of their emotional experiences during game play. Physiological state (heart-rate, galvanic skin response) prediction models have also been proposed for potential entertainment augmentation in computer games [1.30].

Following the theoretical principles reported from Malone [1.9], Koster [1.7] and Yannakakis [1.21], and to a lesser degree from Lazzaro [1.12] and Kline and Arlidge [1.14], this chapter is primarily focused on the contributions of game opponents' behavior (by enabling appropriate learning patterns on which the player may train [1.7]) to the real-time entertainment value of the game. This chapter therefore excludes the socialization factor of entertaining game play and investigates instead entertainment perceived in single player scenarios. We argue that among the three dimensions of "fun" (endurability, engagement, expectations) defined in [1.13] it is only *engagement* that is affected by the opponent since both *endurability* and *expectations* are based primarily on the game design *per se*. Given a successful interactive game design that yields high expectations and endurability, we only focus on the level of engagement that generates fun (entertainment).

Rather than being based purely on theoretical assumptions and visual observations of players' satisfaction, this chapter presents two different approaches that attempt to capture quantitatively the level of player entertainment in computer games. First, a custom-designed quantitative metric of entertainment is proposed, motivated by qualitative considerations of what is enjoyable in computer games. The metric has been validated against humans' preferences. Second, a mapping between estimators of Malone's challenge and curiosity entertainment factors and humans' valuations of entertainment is derived using evolving NNs.

### 1.3 The Test-bed Game

The test-bed studied here is a modified version of the original Pac-Man computer game released by Namco. The player's (*PacMan's*) goal is to eat all the pellets appearing in a maze-shaped stage while avoiding being killed by the four *Ghosts*. The game is over when either all pellets in the stage have been eaten by *PacMan*, *Ghosts* manage to kill *PacMan* or a predetermined number of simulation steps is reached without either of the above occurring. In the last case, the game restarts from the same initial positions for all five characters. In the test-bed game, *PacMan* is controlled by the human player while a multi-layered feedforward neural controller is employed to manage the *Ghosts'* motion.

The game is investigated from the opponents' viewpoint and more specifically how the *Ghosts'* adaptive behaviors and the levels of challenge and curiosity they generate can collectively contribute to player satisfaction. The game field (i.e. stage) consists of corridors and walls. Both dimensions and maze structure of the stage are predefined. For the experiments presented in this chapter we use a  $19 \times 29$  grid maze-stage with corridors 1 grid-cell wide (see [1.31] for more details of the Pac-Man game design).

We choose predator/prey games as the initial genre for this research since, given our aims, they provide us with unique properties: in such games we can deliberately abstract the environment and concentrate on the characters' behavior. Moreover, we are able to easily control a learning process through on-line interaction. Other genres of game (e.g. first person shooters) offer similar properties; however predator/prey games are chosen for their simplicity as far as their development and design are concerned.

### 1.4 Empirical Estimation of Entertainment

As noted in section 1.3, predator/prey games will be our test-bed genre for the investigation of enjoyable games. More specifically, in the games studied, the prey is controlled by the player and the predators are the computer-controlled opponents (non-player characters, or NPCs).

In the approach presented in this section, a quantitative metric of player satisfaction is designed based on general criteria of enjoyment. The first step towards generating enjoyable computer games is therefore to identify the criteria or features of games that collectively produce enjoyment (or else interest) in such games. Second, quantitative estimators for these criteria are defined and combined, in a suitable mathematical formula, to give a single quantity correlated with player satisfaction (interest). Finally, this formulation of player interest is tested against human players' judgement in real conditions using the Pac-Man test-bed (see section 1.4.2).

Following the principles of Yannakakis and Hallam [1.6, 1.22] we will ignore mechanics, audiovisual representation, control and interface contribu-

tions to the enjoyment of the player and we will concentrate on the opponents' behaviors. A well-designed and popular game such as Pac-Man can fulfil all aspects of player satisfaction incorporated in the above-mentioned design game features. The player, however, may contribute to his/her own entertainment through interaction with the opponents of the game and therefore is included implicitly in the interest formulation presented here — see also [1.32] for studies of the player's impact on his/her entertainment.

**Criteria.** By observing the opponents' behavior in various predator/prey games we attempted to identify the key features that generate entertainment for the player. These features were experimentally validated against various opponents with different strategies and redefined when appropriate. Hence, by being as objective and generic as possible, we believe that the criteria that collectively define interest on any predator/prey game are as follows.

1. *When the game is neither too hard nor too easy.* In other words, the game is interesting when predators (opponents) manage to kill the prey (player) sometimes but not always. In that sense, given a specific game structure and a player, highly effective opponent behaviors are not interesting behaviors and *vice versa*.
2. *When there is diversity in opponents' behavior between games.* That is, the game is interesting when NPCs are able to find dissimilar ways of hunting and killing the player in each game so that their strategy is more variable.
3. *When opponents' behavior is aggressive rather than static.* That is, the game is interesting when the predators move constantly all over the game world and cover it uniformly. This behavior gives the player the impression of an intelligent strategy for the opponents.

**Metrics.** These three general criteria must now be expressed in quantitative terms using data observable during game play. We therefore let a group of game opponents — the number of opponents depends on the specific game under examination — play the game  $N$  times (each game for a sufficiently large evaluation period of  $t_{max}$  steps) and record the steps  $t_k$  taken to kill the player in each game  $k$  as well as the total number of visits  $v_{ik}$  opponents make to each cell  $i$  of the game grid.

Given these data, quantifications of the three interest criteria proposed above can be presented as follows.

1. **Appropriate Level of Challenge.** The game is uninteresting when either the opponents consistently kill the player quickly (game too hard) or when the game consistently runs for long periods (game too easy). This criterion can be quantified by  $T$  in (1.1) below.

$$T = [1 - (E\{t_k\}/\max\{t_k\})]^{p_1} \quad (1.1)$$

where  $E\{t_k\}$  is the average number of simulation steps taken to kill the prey-player over the  $N$  games;  $\max\{t_k\}$  is the maximum  $t_k$  over the  $N$

games —  $\max\{t_k\} \leq t_{max}$ ; and  $p_1$  is a weighting parameter.  $T$  is high when  $E\{t_k\}$  is low compared to  $\max\{t_k\}$ , that is, when games occur that are much longer than average.

$p_1$  is adjusted so as to control the impact of the bracketed term in the formula for  $T$ . By selecting values of  $p_1 < 1$  we reward quite challenging opponents more than near-optimal killers, since we compress the  $T$  value toward 1.  $p_1$  is chosen as 0.5 for the experiments presented here.

The  $T$  estimate of interest demonstrates that the greater the difference between the average and the maximum number of steps taken to kill the player, the higher the interest of the game. Given (1.1), both easy-killing ('*too easy*') and near-optimal ('*too hard*') behaviors receive low interest estimate values (i.e.  $E\{t_k\} \simeq \max\{t_k\}$ ). This metric is also called 'challenge'.

- 2. Behavior Diversity.** The game is interesting when the NPCs exhibit a diversity of behavior between games. One manifestation of this is that the time taken to kill the player varies between games. Thus a quantitative metric for this second criterion is given by  $S$  in (1.2) below.

$$S = (\sigma_{t_k}/\sigma_{max})^{p_2} \quad (1.2)$$

where

$$\sigma_{max} = \frac{1}{2} \sqrt{\frac{N}{(N-1)} (t_{max} - t_{min})} \quad (1.3)$$

and  $\sigma_{t_k}$  is the standard deviation of  $t_k$  over the  $N$  games;  $\sigma_{max}$  is an estimate, based on the range of  $\{t_k\}$  of the maximum value of  $\sigma_{t_k}$ ;  $t_{min}$  is the minimum number of steps required for predators to kill the prey when playing against some 'well-behaved' fixed strategy near-optimal predators ( $t_{min} \leq t_k$ ); and  $p_2$  is a weighting parameter which is set so as  $\sigma_{t_k}$  has a linear effect on  $S$  ( $p_2 = 1$ ).

The  $S$  increases proportionally with the standard deviation of the steps taken to kill the player over  $N$  games. Therefore, using  $S$  as defined here, we promote predators that produce high diversity in the time taken to kill the prey.

- 3. Spatial Diversity.** The game is more interesting when opponents appear to move around actively rather than remain static or passively follow the player. A good measure for quantifying this criterion is through entropy of the opponents' visits to the cells of the game grid during a game, since the entropy quantifies the completeness and uniformity with which the opponents cover the stage. Hence, for each game, the cell visit entropy is calculated and normalized into  $[0, 1]$  via (1.4).

$$H_n = \left[ -\frac{1}{log V_n} \sum_i \frac{v_{in}}{V_n} \log \left( \frac{v_{in}}{V_n} \right) \right]^{p_3} \quad (1.4)$$

where  $V_n$  is the total number of visits to all visited cells (i.e.  $V_n = \sum_i v_{in}$ ) and  $p_3$  is a weighting parameter.  $p_3$  is adjusted in order to promote very

high  $H_n$  values and furthermore to emphasize the distinction between high and low normalized entropy values. Appropriate  $p_3$  parameter values which serve this purpose are those greater than one ( $p_3 = 4$  in this chapter), since they stretch the value of  $H_n$  away from 1.

Given the normalized entropy values  $H_n$  for all  $N$  games, the interest estimate for the third criterion can be represented by their average value  $E\{H_n\}$  over the  $N$  games. This implies that the higher the average entropy value, the more interesting the game is.

The three individual criterion metrics defined above are combined linearly to produce a single metric of interest (equation 1.5) whose properties match the qualitative considerations developed above.

$$I = \frac{\gamma T + \delta S + \epsilon E\{H_n\}}{\gamma + \delta + \epsilon} \quad (1.5)$$

where  $I$  is the interest value of the predator/prey game;  $\gamma$ ,  $\delta$  and  $\epsilon$  are criterion weight parameters.

The approach to entertainment modeling represented by equation (1.5) is both innovative and efficient. However, it should be clear from the foregoing discussion that there are many possible formulae such as equation (1.5) which would be consistent with the qualitative criteria proposed for predator/prey games. Other successful quantitative metrics for the appropriate level of challenge, the opponents' diversity and the opponents' spatial diversity may be designed and more qualitative criteria may be inserted in the interest formula. Alternative mathematical functions for combining and weighting the various criteria could be employed.

For example, other metrics for measuring the appropriate level of challenge could be used: one could come up with a  $T$  metric assuming that the appropriate level of challenge follows a Gaussian distribution over  $E\{t_k\}$  and that the interest value of a given game varies depending on how long it is — *very short* ( $E\{t_k\} \approx t_{min}$ ) games tend to be frustrating and *long games* ( $E\{t_k\} \approx max\{t_k\}$ ) tend to be boring. (However, very short games are not frequent in the experiments presented here and, therefore, by varying the weight parameter  $p_1$  in the proposed  $T$  metric (see (1.1)) we are able to obtain an adequate level of variation in measured challenge.)

To obtain values for the interest formula weighting parameters  $\gamma$ ,  $\delta$  and  $\epsilon$  we select empirical values based on the specific game in question. For Pac-Man, spatial diversity of the opponents is of the greatest interest: the game no longer engages the player when *Ghosts* stick in a corner instead of wandering around the stage. Thus, diversity in game play ( $S$ ) and challenge ( $T$ ) should come next in the importance list of interest criteria. Given the above-mentioned statements and by adjusting these three parameters so that the interest value escalates as the opponent behavior changes from randomly generated (too easy) to near-optimal hunting (too difficult) and then to following *Ghost* behaviors, we come up with  $\gamma = 1$ ,  $\delta = 2$  and  $\epsilon = 3$ .

The question remains, however, whether the number produced by such a formula really captures anything useful concerning a notion so potentially complex as human enjoyment. That question is addressed in section 1.4.2 below.

#### 1.4.1 Generality of the Metric

The interest metric introduced in equation (1.5) can be applied in principle to any predator/prey computer game because it is based on generic measurable features of this category of games. These features include the time required to kill the prey and the predators' entropy in the game field. Thus, it appears that (1.5) — or a similar measure based on the same concepts — constitutes a generic interest approximation of predator/prey computer games. Evidence demonstrating the interest metric's generality appears in [1.33] through successful application of the  $I$  value metric to two quite dissimilar predator/prey games.

Moreover, given the two first interest criteria previously defined, the approach can be generalized to all computer games. Indeed, no player likes any computer game that is too hard or too easy to play and, furthermore, any player would enjoy diversity throughout the play of any game. The third interest criterion is applicable to games where spatial diversity is important which, apart from predator/prey games, may also include action, strategy and team sports games according to the computer game genre classification of Laird and van Lent [1.3]. As long as game designers can determine and extract the measurable features of the opponent behavior that generate excitement for the player, and identify observable indices of them that can be computed from data collected during game play, a mathematical formula can be designed in order to collectively represent them.

Finally, a validation experiment like that presented in section 1.4.2 below can be used to assess the performance of the designed formula or test variants of it.

#### 1.4.2 Experiments

Given that the interest measure defined above has been constructed to reflect the designers' intuitions about those features of games that contribute to player interest and satisfaction, one would expect that games with higher values of  $I$  would be judged more satisfying by human players.

To investigate this, the Pac-Man game was used to acquire data on human judgement of entertainment. Thirty subjects (43.3% females, 56.7% males) whose age covered a range between 17 and 51 years participated in this experiment. In addition, all subjects spoke English (language of the survey questionnaire) as a foreign language since their nationality was either Danish (90%) or Greek (10%) [1.21, 1.22].

Subjects in the experiment played against five selected opponents differing in the  $I$  value they generate against a well-behaved computer-programmed player. Each player played several paired sets of games such that all pairwise combinations of the 5 opponents, in each order, were covered by the group of subjects (see [1.6] for details of the experimental design). For each pair, the player indicated which opponent generated the more satisfying game (players were not aware of the computed  $I$  values of the opponents they faced).

The degree of correlation between human judgement of entertainment and the computed  $I$  value is found by matching the entertainment preferences between the five opponents recorded by the human players and the  $I$  value ranking. According to the subjects' answers the  $I$  value is correlated highly with human judgement ( $r = 0.4444$ ,  $p$ -value =  $1.17 \cdot 10^{-8}$  — see [1.21, 1.22]). These five opponents are used as a baseline for validating all entertainment modeling approaches presented in this chapter (see section 1.5.3).

In addition, players completed survey questions designed to elicit information about their likeliness for Pac-Man as a computer game, which allowed them to be grouped into three types: the ones that conceptually did not particularly like Pac-Man, the ones believed that Pac-Man is an interesting game and the ones that liked Pac-Man very much. It was found that neither the type of player nor the order in which opponents were encountered had any significant effect on the player's judgement of the relative interest of the games in a pair (see [1.21, 1.22] for full details of the analysis).

#### 1.4.3 Conclusions

Given observations, intuitions and informal empirical studies on the predator/prey genre of games, generic criteria that contribute to the satisfaction for the player and map to measurable characteristics of the opponent behavior were defined.

According to the hypothesis of Yannakakis and Hallam [1.6], the player-opponent interaction — rather than the audiovisual features, the context, the mechanics, the control, the interface or the genre of the game — is the property that contributes the majority of the quality features of entertainment in a computer game. Given this fundamental assumption, a metric for measuring the real-time entertainment value of predator/prey games was motivated and designed. This value is built upon three generic entertainment criteria: appropriate level of challenge, opponents' behavior diversity and opponents' spatial diversity.

By testing predator/prey games with human players, it was confirmed that the interest value computed by equation (1.5) is consistent with human judgement of entertainment. In fact, the human player's notion of interest in the Pac-Man game seems to correlate highly with the computed interest metric, independently of player type or order of play [1.22].

Thus, we demonstrate the first approach to quantitative modeling of entertainment: the idea of using a custom-designed (but nevertheless quite

generic) mathematical expression that yields a numerical value well-correlated with human judgement. The metric formula rests on measurable features of opponent behavior computable from data collected during game play.

### 1.5 Quantitative Analysis of Entertainment Factors Derived from Psychological studies

The second approach to entertainment capture is in a sense inverse to the empirical, designer-driven, estimation of entertainment presented in section 1.4. In that approach, generic criteria describing interesting games were identified by intuition and consideration of game experiences on the part of the entertainment metric designer.

Although shown to yield a result that correlates well with human judgement, a more satisfying approach might be to start with the theoretical work on qualitative factors of entertainment reviewed in section 1.2 — specifically, Malone’s qualitative factors of entertainment: challenge and curiosity [1.9] — and attempt to construct a quantitative measure using them.

Quantitative measures for challenge and curiosity are inspired by previous work on entertainment metrics [1.6] and computed from data gathered during the real-time player-opponent interaction. The measured challenge and curiosity values are combined with human judgements on the relative entertainment of games using machine learning techniques to construct an expression analogous to equation 1.5 which is highly correlated with human choices. Again, the Pac-Man game is employed as a test-bed for the approach.

Two NN types, namely a feedforward NN and a fuzzy-neural network (fuzzy-NN), are used as alternatives to represent the entertainment metric. In each case, the inputs to the network are the measures of challenge and curiosity for a particular game (opponent). The output is an analogue of the  $I$  metric. The networks are thus used as function approximators for the expression defining the interest metric. Training of the approximators is done using artificial evolution.

The procedures used are described in more detail below. A comparison between the two alternatives is presented and the results are validated against and compared with the custom-designed metric of equation (1.5). The results of the study demonstrate that both NNs represent functions — possible interest metrics based on challenge and curiosity measures — whose qualitative features are consistent with Malone’s corresponding entertainment factors. Furthermore, the evolved feedforward NN provides a more accurate model of player satisfaction for Pac-Man than the custom-designed model (the  $I$  value), presented in the previous section, for this genre of games [1.33].

### 1.5.1 Experimental Data

As a final part of the experiment presented in section 1.4.2, subjects played a set of 25 games against each of two well-behaved opponents ( $A$  and  $B$ ). After the 50 games, the player records whether the first set or the second set of games was the more interesting, i.e. whether  $A$  or  $B$  generated a more interesting game. (The preference of  $A$  over  $B$  is also written as  $A \succ B$ .) To minimize order effects, each subject plays the aforementioned sets both orders during the experiment.

Given the recorded values of human playing times  $t_k$  over the 50 ( $2 \times 25$ ) games against a specific opponent, either  $A$  or  $B$ , the average playing time ( $E\{t_k\}$ ) and the standard deviation of playing times ( $\sigma\{t_k\}$ ) for all subjects are computed. We suggest that the  $E\{t_k\}$  and  $\sigma\{t_k\}$  values are appropriate measures to represent the level of challenge and the level of curiosity respectively [1.9] during game play. The former provides a notion for a goal whose attainment is uncertain (winning the game) — the lower the  $E\{t_k\}$  value, the higher the goal uncertainty and furthermore the higher the challenge — and the latter effectively portrays a notion of unpredictability in the subsequent events of the game — the higher the  $\sigma\{t_k\}$  value the more variable the game duration, so the higher the opponent unpredictability and therefore the higher the curiosity.

### 1.5.2 Tools

Two alternative neural network structures (a feedforward NN and a fuzzy-NN) for learning the relation between the challenge and curiosity factors and the entertainment value of a game have been used and are presented here. The assumption is that the entertainment value  $y$  of a given game is an unknown function of  $E\{t_k\}$  and  $\sigma\{t_k\}$ , which the NN will learn. The subjects' expressed preferences constrain but do not specify the values of  $y$  for individual games. Since there is no *a priori* target  $y$  values for any given game, the output error function is not differentiable, and ANN training algorithms such as back-propagation are inapplicable. Learning is achieved through artificial evolution [1.34] and is described in Section 1.5.2.

**Feedforward NN.** A fully-connected multi-layered feedforward NN has been evolved [1.34] for the experiments presented here. The sigmoid function is employed at each neuron, the connection weights take values from -5 to 5 and both input values are normalized into  $[0, 1]$  before they are entered into the feedforward NN. In an attempt to minimize the controller's size, it was determined that single hidden-layered NN architectures, containing 20 hidden neurons, are capable of successfully obtaining solutions of high fitness (network topology is not evolved, however).

**Fuzzy-NN.** A fuzzy [1.35] Sugeno-style [1.36] inference neural network is trained to develop fuzzy rules by evolving the memberships functions for both the input ( $E\{t_k\}$ ,  $\sigma\{t_k\}$ ) and the output variable  $y$  of the network as well as each fuzzy rule's weight. Each of the input and output values is presented by five fuzzy sets corresponding to very low, low, average, high and very high. The membership functions for the input values are triangular and their center  $\alpha$  and width  $\beta$  are evolved whereas the output fuzzy sets use singleton membership functions [1.36] — only the center  $\alpha$  of the spike membership function is evolved. The centroid technique is used as a defuzzification method.

**Genetic Algorithm.** A generational genetic algorithm (GA) [1.37] is implemented, which uses an “exogenous” evaluation function that promotes the minimization of the difference in matching the human judgement of entertainment. The feedforward NNs and fuzzy-NNs are themselves evolved. In the algorithm presented here, the evolving process is limited to the connection weights of the feedforward NN and the rule weights and membership function parameters of the fuzzy-NN.

The evolutionary procedure used can be described as follows. A population of  $N$  networks is initialized randomly. For feedforward NNs, initial real values that lie within  $[-5, 5]$  for their connection weights are picked randomly from a uniform distribution, whereas for the fuzzy-NNs, initial rule weight values equal 0.5 and their membership function parameter values lie within  $[0, 1]$  (uniformly distributed). Then, at each generation:

Step 1 Each member (neural network) of the population is evaluated with two pairs of ( $E\{t_k\}$ ,  $\sigma\{t_k\}$ ) values, one for  $A$  and one for  $B$ , and returns two output values, namely  $y_{j,A}$  (interest value of the game set against opponent  $A$ ) and  $y_{j,B}$  (interest value of the game set against opponent  $B$ ) for each pair  $j$  of sets played in the survey ( $N_s = 30$ ). If the numerical relationship between  $y_{j,A}$ ,  $y_{j,B}$  matches the preference of subject  $j$  then we state that: ‘the values agree with the subject’ throughout this chapter (e.g.  $y_{j,A} > y_{j,B}$  when subject  $j$  has expressed a preference for  $A$  over  $B$ :  $A \succ B$ ). In the opposite case, we state that: ‘the values disagree with the subject.’

Step 2 Each member  $i$  of the population is evaluated via the fitness function  $f_i$ :

$$f_i = \sum_{j=1}^{N_s} \begin{cases} 1, & \text{if } y_{j,A}, y_{j,B} \text{ agree with subject } j; \\ \left(\frac{1-D(A,B)}{2}\right)^2, & \text{if } y_{j,A}, y_{j,B} \text{ disagree with subject } j. \end{cases} \quad (1.6)$$

where  $D(A, B) = |y_{j,A} - y_{j,B}|$

Step 3 A fitness-proportional scheme is used as the selection method.

Step 4 Selected parents clone an equal number of offspring so that the total population reaches  $N$  members or reproduce offspring by crossover.

The Montana and Davis [1.38] crossover and the uniform crossover operator is applied for feedforward NNs and fuzzy-NNs respectively with a probability 0.4.

Step 5 Gaussian mutation occurs in each gene (connection weight) of each offspring's genome with a small probability  $p_m = 1/n$ , where  $n$  is the number of genes.

The algorithm is terminated when either a good solution (i.e.  $f_i > 29.0$ ) is found or a large number of generations  $g$  is completed ( $g = 10000$ ).

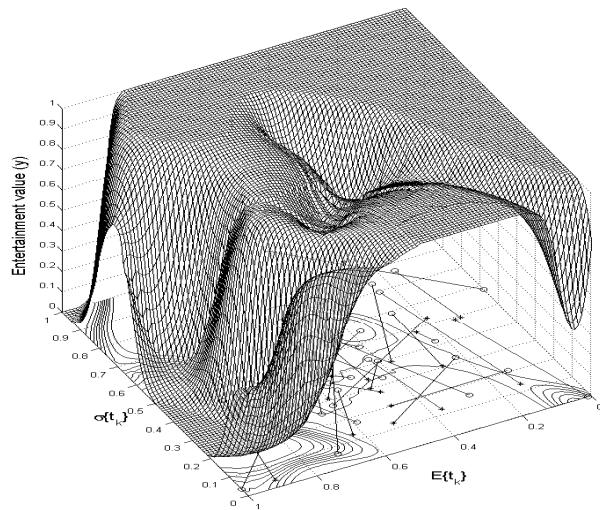
### 1.5.3 Results

Results obtained from both feedforward NN and fuzzy-NN evolutionary approaches are presented in this section. In order to control for the non-deterministic effect of the GA initialization phase, each learning procedure (i.e. GA run) for each NN type is repeated ten times — we believe that this number is adequate to illustrate clearly the behavior of each mechanism — with different random initial conditions.

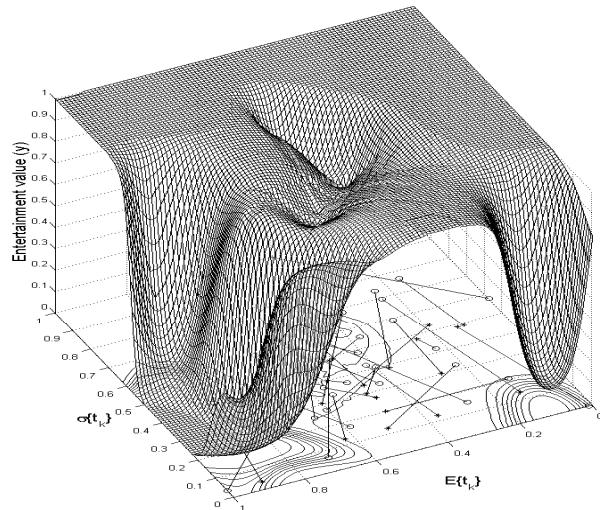
**Evolved Feedforward NN.** For space considerations, only the two fittest solutions achieved from the evolving feedforward NN approach are illustrated in Fig. 1.1. The qualitative features of the surfaces plotted in Fig. 1.1 appeared in all ten learning attempts. The most important conclusions derived from the feedforward NN mapping between  $E\{t_k\}$ ,  $\sigma\{t_k\}$  and entertainment are that:

- Entertainment has a low value when challenge is too high ( $E\{t_k\} \approx 0$ ) and curiosity is low ( $\sigma\{t_k\} \approx 0$ ).
- Even if curiosity is low, if challenge is at an appropriate level ( $0.2 < E\{t_k\} < 0.6$ ), the game's entertainment value is high.
- If challenge is too low ( $E\{t_k\} > 0.6$ ), the game's entertainment value appears to drop, independently of the level of curiosity.
- There is only a single data point present when  $\sigma\{t_k\} > 0.8$  and the generalization of the evolved feedforward NNs within this space appears to be poor. Given that only one out of the 60 different game play data points falls in that region of the two-dimensional ( $E\{t_k\}, \sigma\{t_k\}$ ) space, we can hypothesize that there is low probability for a game to generate curiosity values higher than 0.8. Thus, this region can be safely considered insignificant for these experiments. However, more samples taken from a larger game play survey would be required to effectively validate this hypothesis.

The fittest evolved feedforward NN is also tested against the custom-designed  $I$  metric for cross-validation purposes. The feedforward NN ranks the five different opponents previously mentioned in section 1.4.2 in the order  $I_1 = I_2 < I_4 < I_3 < I_5$  (where  $I_i$  is the entertainment value the  $i$  opponent generates) which yields a correlation of 0.5432 (p-value =  $3.89 \cdot 10^{-12}$ )



**Fig. 1.1a.**The fittest feedforward NN solution ( $f = 29.95$ )



**Fig. 1.1b.**The second fittest feedforward NN solution ( $f = 29.67$ )

**Fig. 1.1a–b.**Circles ('o') and stars ('\*') represent  $E\{t_k\}$ ,  $\sigma\{t_k\}$  values obtained by playing against opponents *A* and *B* respectively. Straight lines are used to connect the sets of games that humans played in pairs

of agreement with human notion of entertainment expressed by the subject choices in the original experiment. Given this ranking of entertainment against these five opponents, the feedforward NN approach appears to model human entertainment better than the custom-designed interest metric proposed in [1.6, 1.22] and described above ( $r = 0.4444$ ,  $p\text{-value} = 1.17 \cdot 10^{-8}$ ).

The relationship between entertainment, challenge and curiosity expressed by the evolved feedforward NNs appears to follow the qualitative principles of Malone's work [1.9] and the human-verified interest metric developed in our previous work [1.6] for predator/prey games. According to these, a game should maintain an appropriate level of challenge and curiosity in order to be entertaining. In other words, too difficult and/or too easy and/or too unpredictable and/or too predictable opponents to play against make the game uninteresting.

#### 1.5.4 Evolving Fuzzy-NN

The evolutionary procedure for the fuzzy-NN approach is also repeated ten times and only the fuzzy-NN that generates the highest fitness ( $f = 29.81$ ) is presented here for reasons of space. Twenty five fuzzy rules are initially designed based the conclusions derived from the evolved feedforward NNs. The fittest fuzzy-NN generates 19 fuzzy rules in total — rules with weight values less than 0.1 are not considered significant and therefore are excluded from further consideration — which are presented here with their corresponding weight values  $w$ :

- Entertainment is *very low* if (a) challenge is *very high* and curiosity is *low* (Rule 1;  $w_1 = 0.4440$ ) and (b) challenge is *low* and curiosity is *average* (Rule 2;  $w_2 = 0.3617$ ).
- Entertainment is *low* if (a) challenge is *very low* and curiosity is *average* (Rule 3;  $w_3 = 0.9897$ ) or *low* (Rule 4;  $w_4 = 0.7068$ ); (b) challenge is *low* and curiosity is *high* (Rule 5;  $w_5 = 0.7107$ ); (c) challenge is *high* and curiosity is *very low* (Rule 6;  $w_6 = 0.5389$ ) and (d) challenge is *very high* and curiosity is *very low* (Rule 7;  $w_7 = 0.9520$ ) or *high* (Rule 8;  $w_8 = 0.9449$ ).
- Entertainment is *average* if challenge is *very low* and curiosity is *high* (Rule 9;  $w_9 = 0.5818$ ).
- Entertainment is *high* if (a) challenge is *low* and curiosity is *very low* (Rule 10;  $w_{10} = 0.8498$ ) or *very high* (Rule 11;  $w_{11} = 0.2058$ ); (b) challenge is *average* and curiosity is *low* (Rule 12;  $w_{12} = 0.5$ ); (c) challenge is *high* and curiosity is *low* (Rule 13;  $w_{13} = 0.2824$ ) or *average* (Rule 14;  $w_{14} = 0.25$ ) and (d) challenge is *very high* and curiosity is *average* (Rule 15;  $w_{15} = 0.2103$ ).
- Entertainment is *very high* if (a) challenge is *very low* and curiosity is *very high* (Rule 16;  $w_{16} = 0.7386$ ); (b) challenge is *average* and curiosity is *very low* (Rule 17;  $w_{17} = 0.5571$ ) or *very high* (Rule 18;  $w_{18} = 0.8364$ ) and (c) challenge is *high* and curiosity is *high* (Rule 19;  $w_{19} = 0.2500$ ).

The quantitative measures of entertainment achieved through the neuro-fuzzy approach and the majority of the fuzzy rules generated appear consistent with Malone's principles of challenge and curiosity, the empirical contributions of the interest metric from the literature [1.22] and the fittest feedforward NN presented in section 1.5.3. However, the fittest fuzzy-NN (being less fit than the fittest feedforward NN) generates some few fuzzy rules that are not consistent with the aforementioned principles — e.g. Rule 10: entertainment is *high* if challenge is *low* and curiosity is *very low*. It is not clear whether the poorer performance is intrinsic to the method or a result of unlucky initialization; further tests are needed to distinguish these alternatives.

The fuzzy-NN is tested against the  $I$  metric of section 1.4 as in the evolved feedforward NN approach. The evolved fuzzy-NN ranks the five opponents in the order  $I_2 < I_1 < I_3 < I_4 = I_5$ . This ranking demonstrates a correlation of 0.3870 (p-value =  $1.74006 \cdot 10^{-6}$ ) of agreement with human notion of entertainment, which appears to be lower than the correlation achieved through the  $I$  value of section 1.4 ( $r = 0.4444$ , p-value =  $1.17 \cdot 10^{-8}$  [1.6]). However, as in the feedforward NN approach, the generalization of the evolved fuzzy-NNs appears to be poor when  $\sigma\{t_k\} > 0.8$  due to the presence of a single data point within this region of the  $(E\{t_k\}, \sigma\{t_k\})$  space. Even though we consider this non-frequent region as insignificant as far as this work is concerned, it may be sampled from a more extensive human game experiment in a future study.

### 1.5.5 Conclusions

This section introduced an alternative approach to constructing a quantitative metric of entertainment motivated by the qualitative principles of Malone's intrinsic factors for engaging game play [1.9]. More specifically, the quantitative impact of the factors of challenge and curiosity on human entertainment were investigated in the Pac-Man game.

The two neuro-evolution approaches for modeling entertainment examined demonstrate qualitative features that share principles with the interest metric ( $I$  value) presented in section 1.4. This (second) approach replaces the hand-crafted mathematical formulation of the interest metric with a more general process of machine learning applied to neural network models. Both obtained models manage to map successfully between the measures of entertainment factors such as challenge and curiosity and the notion of human game play satisfaction.

Validation results obtained show that the fittest feedforward NN gets closer — in the sense of statistical correlation — to the human notion of entertainment than both the  $I$  value [1.22] and the fittest fuzzy-NN. Therefore, it appears that the average and the standard deviation of a human's playing time over a number of games are in themselves adequate, and in fact

more effective than the  $I$  value (as reported in [1.22]), for capturing player entertainment in real-time in predator/prey games.

The reported work on this approach is most significantly limited by the number of participants in the game survey we devised. Therefore, not all regions of the challenge-curiosity search space were sampled by human play which therefore yielded poor NN generalization for these regions. Limited data also restricted the sensible number of inputs to the learning system.

Malone's entertainment factor of fantasy is omitted here since the focus is on the contribution of the opponent behaviors to the generation of entertainment; however, experiments on interactive physical predator/prey games with children have shown that entertainment increases monotonically with respect to the fantasy factor [1.39].

This second entertainment modeling approach presented here demonstrates generality over the majority of computer game genres since the quantitative means of challenge and curiosity are estimated through a generic feature of game play which is the playing time of humans over a number of games. Thus, these or similar measures could be used to measure player satisfaction in any genre of game. However, each game possesses additional idiosyncratic entertainment features that might need to be extracted and added to the proposed generic measures used as input to the machine learning tools — therefore, more games of the same or other genres need to be tested to evaluate the true generality of this approach.

## 1.6 Discussion

The foregoing has proposed and demonstrated a pair of methods for deriving a quantitative estimate of the level of entertainment experienced by a player of a computer game, using data that can be derived from or during game play. In this section, we discuss some of the questions raised by the approach and the assumptions on which it is based.

An immediate and natural question is whether the techniques described really capture “entertainment” which, after all, is a complex mental phenomenon depending on the player, the game and (probably) a number of external factors in rather involved ways. We acknowledge that the definition of a quantitative metric for entertainment in this sense is almost certainly infeasible. However, we take a practical approach here: it is sufficient for our purposes if a quantity exists that can be computed from observable data from the player-game interaction and that correlates well with players' expressed preferences. In other words, a numerical value which orders games in the same way as players' judgement of entertainment is sufficient for our purposes.

The foregoing material illustrates two ways to construct such a metric: by design, using the insights of a skilled game player; and by using machine learning to explore the space of possible evaluation functions whose values are consistent with human judgement of entertainment value. The resulting

metric is specific to the particular game under consideration, but the general method of construction is applicable to a wide variety of game instances and genres.

To summarize, therefore, the proposed approach does not capture details of the complex mental states associated with enjoyment of computer games, but it does provide a way to evaluate different instances of game play in terms of how entertaining they are for a player. Such knowledge can be used, for example, for tuning the game to suit the player (see below).

In addition to this general question concerning the approach, there are a number of assumptions (and hence limitations) associated with the methods presented; these are discussed below.

### 1.6.1 Assumptions and Limitations of the $I$ value

The interest metric described in section 1.4 is based on specific assumptions about the features of a game that generate enjoyment for the player.

- The approach assumes that interaction with opponent behavior is the primary source of variability in the entertainment value of a given game. That is, the enjoyment generated by the graphical, multimedia and storyline components of the game design is disregarded. This is a reasonable assumption for comparisons between instances of play of a single given game, but means that the interest metric is specific to a certain game and cannot be used for meaningful comparison between different games (e.g. to answer “Is Space Invaders more entertaining than Quake?”).
- The interest metric is calculated using data obtained from a sample of  $N$  games. This is consistent with human cognition since it appears that human players require a significant number of games (or else playing time) to classify a specific computer game according to their perceived satisfaction. However, this assumption constitutes a limitation of the method. A further investigation of the relationship between the  $I$  value and the  $N$  played games might reveal that fewer games are needed for an estimate that is still consistent with human notion of perceived entertainment.
- The interest value definition assumes that players of the game have average-level playing skills. By ‘average-level’ we only exclude the two following extreme player types: (1) those who have never played the specific game before and furthermore do not know the rules and how to play it — these players perform poorly against almost any type of opponent; (2) those who have an excellent knowledge of the specific game, can easily predict the opponent behavior and have mastered the game controls. These players can usually beat even the most effective opponents designed for the game. In each case, the interest value might not be very well estimated since the challenge criterion  $T$  approximates a zero value regardless of the opponent, in the first case because the game is much too hard and in the second because it is too easy (recall that  $T$  is designed to be maximum for a

‘reasonably difficult’ game). This appears to be consistent with human notion of interestingness since we believe that neither extreme player type will find the game interesting.

- The interest value depends primarily on the opponents’ behavior. Implicitly, through that, it depends on the player’s behavior since the opponent behavior is elicited in interaction with the player. (The previous point concerning playing skills is a specific instance of this more general observation). If the players of a game can be divided into classes with quite different playing styles , for example “aggressive” vs. “defensive”, then it may be necessary to design a separate  $I$  metric formula for each player type, because of the difference in opponent behavior elicited by their differing styles of play. For a comprehensive discussion of this assumption in [1.32] where the interest value dependence on the player is investigated through experiments in the Pac-Man test-bed game.
- A factor that may contribute to enjoyable games is the match between the real-time speed of the game and the reaction time of the player. Gradually decreasing the required player reaction time is a standard and inexpensive technique used by a set of games (i.e Pac-Man) to achieve increasing challenge for the player as the game proceeds. This is not considered in the work in this chapter since changing the demanded reaction time does not alter the qualitative properties of the opponent behavior (except through the implicit dependence on the player’s style of play). Note that in the extreme case, an unintelligent opponent may generate an exciting game just because of the unrealistically fast reaction time required of the player.

### **1.6.2 Assumptions and Limitations of the Machine Learning Approach**

The second approach to constructing a metric uses machine learning rather than designer insight to build a quantitative evaluation of a game. This method is based on the same fundamental assumptions as the  $I$  value approach: that the opponents’ behavior is the primary determinant of the entertainment value of a given instance of game play. Many of the comments of the previous section also apply to this approach. However, there are a few observations specific to this methodology.

- The issue of playing style is arguably less relevant here than in the designer-insight method. If it is necessary to determine player type to be able to evaluate games, then the requirement of a consistent metric will in principle force the machine learning technique to perform an implicit player-type classification, assuming that the function representation technology (here, a neural network) is powerful enough to do so. In other words, because this approach can construct much more complex (and therefore less scrutable) mappings from raw measurements to entertainment metric value, it can

cope with more complex relationships between the data and the metric than a designer is likely to be able to manage.

- However, the effectiveness of a machine learning method depends strongly on the quantity and quality of data available. For the case considered here, this data comprises two kinds: raw measurements derived from game play, that represent aspects of features such as challenge, curiosity, fantasy, etc.; and expressed preferences or rankings between instances of game play. The former provide the observables on which the metric is built, the latter determine the degree of consistency with human judgement of any given proposal for the metric function.
- Obtaining the latter kind of data involves experimentation in which players are asked to say which of several (here two) instances of game play they prefer; collecting such data is time- and player- consuming. This limits the complexity of metric that can be considered, since it limits the feedback available to the machine learning process during the exploration of the space of metrics.
- The former kind of data also presents certain problems: specifically, how do we know what measurements to include as a basis for the metric? The work presented here uses one designer-chosen measurement for each relevant feature — challenge and curiosity — but one could in principle devise many measurements correlated with either feature, and allow the machine learning system to use all of them or to make a selection of the best measurements for the purpose. This approach removes a certain designer bias in the method at the expense of complicating the machine learning task and approaching earlier the limits imposed by the difficulty of collecting preference data from players.
  - The issue of what value of  $N$  to choose can be finessed in this second approach, as can the question of game speed and demanded player reaction time, by appropriate choice of measurements from which to build the evaluation metric. For instance, game speed can be included directly as a measurement and measurements requiring different numbers of games to compute can be included in the process.

### 1.6.3 Making Use of Entertainment Metrics

Given a successful metric of entertainment for a given game, designed and evaluated using one of the methods proposed above, the final question we consider here is how such knowledge might be used. As previously noted, opponents which can learn and adapt to new playing strategies offer a richer interaction to entertain the player. An obvious use of the entertainment metric is therefore to adapt the game so that the metric value increases.

Two possible strategies for this might be:

- to use a machine learning mechanism for the game studied which allows opponents to learn while playing against the player (i.e. on-line). The entertainment metric can be used to guide the learning process.

Such an on-line learning mechanism is comprehensively described in [1.6, 1.33, 1.22]. Its basic steps are presented briefly here as follows. At each generation of the algorithm:

- Step 1: Each opponent is evaluated every  $e_p$  simulation steps via an individual reward function that provides an estimate of the  $I$  value of the game, while the game is played.
- Step 2: A pure elitism selection method is used where only a small percentage of the fittest opponents is able to breed. The fittest parents clone offspring.
- Step 3: Mutation occurs in each opponent (varying neural network controller connection weights) with a suitable small probability.
- Step 4: Each mutated offspring is evaluated briefly in off-line mode, that is, by replacing the least-fit member of the population and playing a short off-line game of  $e_p$  simulation steps against a selected computer-programmed player. The fitness values of the mutated offspring and the least-fit member are compared and the better one is deployed in the game.

The algorithm is terminated when a predetermined number of games has been played or a game of high interest (e.g.  $I \geq 0.7$ ) is found.

Results reported in [1.21, 1.22] demonstrate that 50 on-line learning games are not enough (in Pac-Man) for the on-line learning mechanism to cause a difference in the  $I$  value which is noticeable by human players. The on-line learning period of 50 games is an empirical choice to balance efficiency and experimental time. The duration of the on-line learning procedure in this experiment lasted 20 minutes on average while the whole human survey experiment presented in section 1.4.2 and section 1.5.1 exceeded 65 minutes in many cases, which is a great amount of time for a human to concentrate.

- to use a metric evaluation function constructed using the machine learning technique directly to enhance the entertainment provided by the game. The key to this is the observation that the models (feedforward NN or fuzzy-NN) relate game features to entertainment value. It is therefore possible in principle to infer what changes to game features will cause an increase in the interestingness of the game, and to adjust game parameters to make those changes. For the feedforward NN, the partial derivatives of  $\partial y / \partial E\{t_k\}$  and  $\partial y / \partial \sigma\{t_k\}$  indicate the change in entertainment for a small change in an individual game feature. One could use gradient ascent to attempt to improve entertainment with such a model. The fuzzy-NN approach provides qualitative rules relating game features to entertainment, rather than a quantitative function, but an analogous process could be applied to augment game entertainment.

Such a direction constitutes an example of future work within computer, physical and educational games. The level of engagement or motivation of the user/player/gamer of such interactive environments can be increased by

the use of the presented approaches providing systems of richer interaction and qualitative entertainment [1.5],

## Acknowledgements

This work was supported in part by the Danish Research Agency, Ministry of Science, Technology and Innovation (project no: 274-05-0511).

## Resources

### Key Books

- Koster, R., A Theory of Fun for Game Design, Paraglyph Press, 2005.

### Key Papers

- Andrade, G., Ramalho, G., Santana, H., Corruble, V. “Challenge-Sensitive Action Selection: an Application to Game Balancing,” in *Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT-05)*, pp. 194–200, Compiègne, France. IEEE Computer Society, 2005.
- Iida, H., Takeshita, N., Yoshimura, J., “A metric for entertainment of boardgames: its implication for evolution of chess variants,” in *Nakatsu, R., Hoshino, J., eds.: IWEC2002 Proceedings*, pp. 65–72, Kluwer, 2003.
- Malone, T. W., “What makes computer games fun?,” *Byte*, vol. 6, pp. 258–277, 1981.
- Lazzaro, N., “Why we play games: Four keys to more emotion without story,” Technical report, XEO Design Inc. 2004.
- Yannakakis, G. N., Hallam, J., “Evolving Opponents for Interesting Interactive Computer Games,” in *Proceedings of the 8th International Conference on the Simulation of Adaptive Behavior (SAB’04); From Animals to Animats 8*, pp. 499–508, Los Angeles, CA, USA, July 13–17, 2004. The MIT Press.
- Yannakakis, G. N., Hallam, J., “Towards Optimizing Entertainment in Computer Games,” *Applied Artificial Intelligence*, 2007 (to appear).

### Discussion Groups, Forums

- Optimizing player satisfaction in games discussion group:  
<http://groups.google.com/group/optimizing-player-satisfaction-in-games>
- Player experience special interest group of DIGRA; list url:  
<http://mail.digra.org/mailman/listinfo/player-experience>

## Key International Conferences/Workshops

Workshop series on Optimizing Player Satisfaction. In conjunction with

- Simulation of Adaptive Behaviour (SAB) Conference, Rome, Italy, in 2006 and
- Artificial Intelligence and Interactive Digital Entertainment (AIIDE) Conference, Stanford, US, in 2007.

## References

- 1.1 Champandard, A.J.: AI Game Development. New Riders Publishing (2004)
- 1.2 Funge, J.D.: Artificial Intelligence for Computer Games. A. K. Peters Ltd, (Wellesley, Massachusetts, USA)
- 1.3 Laird, J.E., van Lent, M.: Human-level AI's killer application: Interactive computer games. In: Proceedings of the Seventh National Conference on Artificial Intelligence (AAAI). (2000) 1171–1178
- 1.4 Vorderer, P., Hartmann, T., Klimmt, C.: Explaining the enjoyment of playing video games: the role of competition. In Marinelli, D., ed.: ICEC conference proceedings 2003: Essays on the future of interactive entertainment, Pittsburgh, (Carnegie Mellon University Press) 107–120
- 1.5 Yannakakis, G.N., Hallam, J.: A scheme for creating digital entertainment with substance. In: Proceedings of the Workshop on Reasoning, Representation, and Learning in Computer Games, 19th International Joint Conference on Artificial Intelligence (IJCAI). (2005) 119–124
- 1.6 Yannakakis, G.N., Hallam, J.: Evolving Opponents for Interesting Interactive Computer Games. In Schaal, S., Ijspeert, A., Billard, A., Vijayakumar, S., Hallam, J., Meyer, J.A., eds.: From Animals to Animats 8: Proceedings of the 8<sup>th</sup> International Conference on Simulation of Adaptive Behavior (SAB-04), Santa Monica, LA, CA, The MIT Press (2004) 499–508
- 1.7 Koster, R.: A Theory of Fun for Game Design. Paraglyph Press (2005)
- 1.8 Kapoor, A., Mota, S., Picard, R.: Towards a Learning Companion that Recognizes Affect. In: Proceedings of Emotional and Intelligent II: The Tangled Knot of Social Cognition, AAAI Fall Symposium . (2001)
- 1.9 Malone, T.W.: What makes computer games fun? Byte **6** (1981) 258–277
- 1.10 Csikszentmihalyi, M.: Flow: The Psychology of Optimal Experience. New York: Harper & Row (1990)
- 1.11 Sweetser, P., Wyeth, P.: GameFlow: A Model for Evaluating Player Enjoyment in Games. ACM Computers in Entertainment **3** (2005)
- 1.12 Lazzaro, N.: Why we play games: Four keys to more emotion without story. Technical report, XEO Design Inc. (2004)
- 1.13 Read, J., MacFarlane, S., Cassey, C.: Endurability, engagement and expectations. In: Proceedings of International Conference for Interaction Design and Children. (2002)
- 1.14 Kline, S., Arlide, A.: Online Gaming as Emergent Social Media: A Survey. Technical report, Media Analysis Laboratory, Simon Fraser University (2003)
- 1.15 Crispini, N.: Considering the growing popularity of online games: What contributes to making an online game attractive, addictive and compelling. Dissertation, SAE Institute, London (2003)

- 1.16 Choi, D., Kim, H., Kim, J.: Toward the construction of fun computer games: Differences in the views of developers and players. *Personal Technologies* **3** (1999) 92–104
- 1.17 Yannakakis, G.N., Lund, H.H., Hallam, J.: Modeling Children’s Entertainment in the Playware Playground. In: Proceedings of the IEEE Symposium on Computational Intelligence and Games, Reno, USA, IEEE (2006) 134–141
- 1.18 Nareyek, A.: Intelligent agents for computer games. In Marsland, T., Frank, I., eds.: *Computers and Games*, Second International Conference, CG 2002. (2002) 414–422
- 1.19 Taatgen, N.A., van Oploo, M., Braaksma, J., Niemantsverdriet, J.: How to construct a believable opponent using cognitive modeling in the game of set. In: Proceedings of the fifth international conference on cognitive modeling. (2003) 201–206
- 1.20 Iida, H., Takeshita, N., Yoshimura, J.: A metric for entertainment of boardgames: its implication for evolution of chess variants. In Nakatsu, R., Hoshino, J., eds.: IWEC2002 Proceedings, Kluwer (2003) 65–72
- 1.21 Yannakakis, G.N.: AI in Computer Games: Generating Interesting Interactive Opponents by the use of Evolutionary Computation. Ph.d. thesis, University of Edinburgh (2005)
- 1.22 Yannakakis, G.N., Hallam, J.: Towards Optimizing Entertainment in Computer Games. *Applied Artificial Intelligence* (2007) to appear.
- 1.23 Yannakakis, G.N., Hallam, J.: Towards Capturing and Enhancing Entertainment in Computer Games. In: Proceedings of the 4<sup>th</sup> Hellenic Conference on Artificial Intelligence, Lecture Notes in Artificial Intelligence. Volume 3955., Heraklion, Greece, Springer-Verlag (2006) 432–442
- 1.24 Andrade, G., Ramalho, G., Santana, H., Corruble, V.: Extending reinforcement learning to provide dynamic game balancing. In: Proceedings of the Workshop on Reasoning, Representation, and Learning in Computer Games, 19th International Joint Conference on Artificial Intelligence (IJCAI). (2005) 7–12
- 1.25 Verma, M.A., McOwan, P.W.: An adaptive methodology for synthesising Mobile Phone Games using Genetic Algorithms. In: Congress on Evolutionary Computation (CEC-05), Edinburgh, UK (2005) 528–535
- 1.26 Hunicke, R., Chapman, V.: AI for Dynamic Difficulty Adjustment in Games. In: Proceedings of the Challenges in Game AI Workshop, 19<sup>th</sup> Nineteenth National Conference on Artificial Intelligence (AAAI’04). (2004)
- 1.27 Spronck, P., Sprinkhuizen-Kuyper, I., Postma, E.: Difficulty Scaling of Game AI. In: Proceedings of the 5th International Conference on Intelligent Games and Simulation (GAME-ON 2004). (2004) 33–37
- 1.28 Yannakakis, G.N., Hallam, J., Lund, H.H.: Capturing Entertainment through Heart-rate Dynamics in the Playware Playground. In: Proceedings of the 5<sup>th</sup> International Conference on Entertainment Computing, Lecture Notes in Computer Science. Volume 4161., Cambridge, UK, Springer-Verlag (2006) 314–317
- 1.29 Rani, P., Sarkar, N., Liu, C.: Maintaining optimal challenge in computer games through real-time physiological feedback. In: Proceedings of the 11<sup>th</sup> International Conference on Human Computer Interaction. (2005)
- 1.30 McQuiggan, S., Lee, S., Lester, J.: Predicting User Physiological Response for Interactive Environments: An Inductive Approach. In: Proceedings of the 2<sup>nd</sup> Artificial Intelligence for Interactive Digital Entertainment Conference. (2006) 60–65
- 1.31 Yannakakis, G.N., Hallam, J.: A generic approach for generating interesting interactive pac-man opponents. In Kendall, G., Lucas, S., eds.: *Proceedings*

- of the IEEE Symposium on Computational Intelligence and Games, Essex University, Colchester, UK (2005) 94–101
- 1.32 Yannakakis, G.N., Maragoudakis, M.: Player modeling impact on player's entertainment in computer games. In: Proceedings of the 10<sup>th</sup> International Conference on User Modeling; Lecture Notes in Computer Science. Volume 3538., Edinburgh, Springer-Verlag (2005) 74–78
- 1.33 Yannakakis, G.N., Hallam, J.: A Generic Approach for Obtaining Higher Entertainment in Predator/Prey Computer Games. *Journal of Game Development* **1** (2005) 23–50
- 1.34 Yao, X.: Evolving artificial neural networks. In: Proceedings of the IEEE. Volume 87. (1999) 1423–1447
- 1.35 Zadeh, L.: Fuzzy sets. *Information and Control* **8** (1965) 338–353
- 1.36 Sugeno, M.: Industrial Applications of Fuzzy Control. North-Holland (1985)
- 1.37 Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor, MI (1975)
- 1.38 Montana, D.J., Davis, L.D.: Training feedforward neural networks using genetic algorithms. In: Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89), San Mateo, CA, Morgan Kauffman (1989) 762–767
- 1.39 Yannakakis, G.N., Hallam, J., Lund, H.H.: Comparative Fun Analysis in the Innovative Playware Game Platform. In: Proceedings of the 1<sup>st</sup> World Conference for Fun 'n Games. (2006) 64–70

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/225182857>

# Dynamic Game Balancing by Recognizing Affect

Chapter · October 2008

DOI: 10.1007/978-3-540-88322-7\_9 · Source: DBLP

---

CITATIONS

38

READS

481

3 authors:



Tim Tijs

Saxion University of Applied Sciences

6 PUBLICATIONS 1,083 CITATIONS

[SEE PROFILE](#)



Dirk Brokken

37 PUBLICATIONS 963 CITATIONS

[SEE PROFILE](#)



Wijnand A Ijsselsteijn

Eindhoven University of Technology

280 PUBLICATIONS 9,554 CITATIONS

[SEE PROFILE](#)

Some of the authors of this publication are also working on these related projects:



Restorative Environments and Healing Media [View project](#)



Empathy & Professional Skill Development in e-Mental Health [View project](#)

# Dynamic Game Balancing by Recognizing Affect

Tim J.W. Tijs<sup>1</sup>, Dirk Brokken<sup>2</sup>, and Wijnand A. IJsselsteijn<sup>3</sup>

<sup>1</sup> User-System Interaction Program, Department of Industrial Design  
Eindhoven University of Technology, P.O. Box 513, 5600MB, Eindhoven, The Netherlands

[t.j.w.tijs@tue.nl](mailto:t.j.w.tijs@tue.nl)

<sup>2</sup> Philips Research Laboratories Europe  
High Tech Campus 34, 5656AE, Eindhoven, The Netherlands  
[dirk.brokken@philips.com](mailto:dirk.brokken@philips.com)

<sup>3</sup> Human-Technology Interaction Group, Department of Technology Management  
Eindhoven University of Technology, P.O. Box 513, 5600MB, Eindhoven, The Netherlands  
[w.a.ijsselsteijn@tue.nl](mailto:w.a.ijsselsteijn@tue.nl)

**Abstract.** Dynamic game balancing concerns changing parameters in a game to avoid undesired player emotions, such as boredom and frustration. This is e.g. done by adjusting the game's difficulty level to the (increasing) skill level of the player during the game. Currently, most balancing techniques are based on in-game performance, such as the player's position in a race. This is, however, insufficient since different types of players exist, with different goals, preferences and emotional responses. Therefore, to deal effectively with a player's emotions, a game needs to look beyond the player's performance. This paper provides an overview of two groups of potentially useful sources for dynamic game balancing: Overt behavior and physiological responses. In addition, we present EMO-Pacman, a design case that aims to implement these new balancing techniques into the game Pac-Man.

**Keywords:** Computer games, emotionally adaptive games, game balancing, affective loop, psychophysiology, emotions.

## 1 Introduction

To be enjoyable, a computer game must be balanced well. Adams & Rollins [1] list a number of requirements for a well-balanced game. For instance, a game must provide meaningful choices, the role of chance should not be so great that player skills become irrelevant, and players must perceive the game to be fair. Concerning the fairness of a (player-versus-environment) game, it is of key-importance that the game's difficulty is adjusted to the player's abilities. This is what many believe to be at the core of game balancing: Changing parameters in order to avoid undesired player emotions such as frustration (because the game is too hard) or boredom (because the game is too easy) [2]. Since the player's abilities tend to increase throughout the game, the game's difficulty level should be adapted continuously. Hence the need for dynamic game balancing.

Currently, a number of heuristic techniques are used for dynamic game balancing, which try to assess the game's perceived difficulty level given a particular game state. Examples of these heuristics are I) the rate of successful shots or hits, II) number of life points, III) time left to complete a task and IV) the player's current position in a race. Regarding the latter heuristic, a commonly applied adaptation technique is rubber banding [3]: When falling behind, the player suddenly gets a boost in speed, which allows for catching up again (and/or the competing cars are slowed down).

However, game adaptation that is solely based on in-game performance can only have limited success, because there are many different types of players [4]. Each type of player has his/her own goals, preferences and emotional responses when playing a game. Therefore, taking the emotional (or affective) state of the player into account is expected to increase interest and fun; games can become emotionally adaptive. Results and approaches from research disciplines such as psychophysiology and affective computing can help in realizing this.

## 2 Emotionally Adaptive Games

Three types of methods for measuring affect can be distinguished [5]: Through self-reporting, by analyzing overt behavior and by analyzing physiological responses. For game balancing in a real-time environment, self-reporting is too obtrusive. However, self-reporting can be used to validate the results of the other two types. Regarding overt behavior<sup>1</sup>, a number of potentially interesting techniques have previously been studied in an affective computing context. Techniques and studies focused on overt behavior that seem particularly useful for dynamic game balancing are provided in Table 1. A more detailed overview can be found at [6]. The relations between affect and physiological responses are investigated in the field of psychophysiology. A number of relevant techniques and studies can be found in Table 1; more detailed overviews can be found in e.g. [7] and [8].

**Table 1.** Potential sources for real-time analysis of affect

---

### Overt behavior / expressions

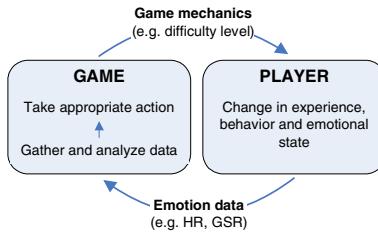
Posture analysis, force exerted on game controls, facial emotions, gestures analysis, speech analysis.

### Physiological responses

Heart rate (HR), Heart rate variability (HRV), Respiration rate (RSPRATE), Coherence between respiration rate and heart rate (RSP-HR\_COH), Blood pressure, Blood volume pulse (BVP), Activity of the corrugator supercilii muscle, (CORR: for moving the eyebrows), Activity of the zygomaticus major muscle, (ZYG: for moving the cheeks), Activity of the orbicularis oculi muscle (OO: for moving the eye-lids), Skin conductance level (SCL), Skin conductance responses (SCR), Eye movements (EOG), Pupil size, Eye blink rate, Brain activity at various frequencies (EEG), Evoked Response Potential (ERP).

---

<sup>1</sup> In this category, we also include behavioral phenomena that are less overt, such as keyboard force (which is measured through sensors).



**Fig. 1.** The emotionally adaptive games loop, inspired on [9] and [16]

From an emotion-perspective, effective human-computer interaction can be realized using an "affective loop" [9]. Fig. 1 shows an affective loop that is tailored to the games domain. By providing the right game mechanics (e.g. audiovisuals, narrative, challenge), the game influences the player's experience, behavior and emotional state. During play, the emotional state of the player (measured in terms of *emotion-data*, such as the types mentioned in Table 1), is continuously being fed back to the game so that the game can adapt its mechanics (e.g. difficulty level) accordingly in real-time. All of this is done with the aim to optimize the experience.

Previous research attempts to create emotionally adaptive software have mainly focused on tutoring systems / productivity software (see e.g. [10]). Fewer attempts have been made to incorporate a closed-loop mechanism in a games context. Several authors [11, 12] have created games that improve player performance by adapting the difficulty level to the player's **physiological state**. Concept validation claims of these two studies were, however, based on a limited number of subjects. Besides these attempts, a number of **biofeedback** games have recently been developed, integrating the player's physiological data into the game (e.g. [13,14]). These games focus on **stress manipulation rather than optimization of gameplay experience**. Closest to the concept described in Fig. 1 probably is the work by Saari and colleagues. They created the Mind-Based Technology framework for psychological customization [15] and, based on this, initiated a framework for emotionally adaptive games (e.g. [16]).

To obtain empirical support for the emotionally adaptive games concept, an emotionally adaptive version of the game Pac-Man is currently being developed in the EMO-Pacman project, as described in the next section of this paper.

### 3 EMO-Pacman

EMO-Pacman (explained in more detail in [17]) is a two-stage attempt to create a game (adapted from Pac-Man, [18]) that adjusts its speed to the player's current emotional state. In the first stage of the project, we investigated the relations between the individual elements of the emotionally adaptive games framework (Fig. 1). The main research question in this stage was "**What game mechanic setting causes what kind of emotional state, and what emotion-data is this accompanied by?**"

To investigate this, a user test was conducted involving 24 adult subjects. One game mechanic, **game speed**, was manipulated, consisting of 3 levels (referred to as slow-mode, fast-mode and normal-mode). Changes in speed affected both the player's character (Pac-Man) and his/her opponents (ghosts). Speed was selected as an

**Table 2.** Effects of game speed on emotion-data

| <b>Significant (<math>p &lt; 0.05</math>) results</b> |  |
|---|--|
| <b>Feature</b>  | <b>Speed mode identified (desired action)</b>      |
| Mean SCL  | slow (speed up)                                    |
| Number of SCR   | slow (speed up)                                    |
| Mean HR   | slow (speed up)                                    |
| Mean RSPRATE  | slow (speed up)                                    |
| Mean ampl. of the CORR signal                         | fast (slow down)                                   |
| Mean ampl. of the ZYG signal                          | fast (slow down)                                   |
| Mean key-press force                                  | slow (speed up) & fast (slow down) & normal (none) |

| <b>No significant results</b>  |
|--|
| Mean value of the CORR signal, mean value of the ZYG signal, mean value of the BVP signal, mean ampl. of BVP signal, mean RSP-HR_COH, HRV: mean power perc. of the 0.04-0.15Hz spectrum. |

independent variable since it is relatively simple to manipulate, and speed changes were expected to have a strong influence on the player's emotional state. Different from the original Pac-Man version, the objective was not to stay alive and finish all levels. Instead, the objective was to score as many points. The player gained points by letting Pacman eat objects (scared ghosts, points, fruit) and lost points when Pacman was eaten by one of his opponents (angry ghosts). During the test, the game was paused twice to let the player report on their emotional state (bored / frustrated / enjoyed). In parallel, a series of emotion-data features was recorded during game play (see Table 2), explained in more detail in [17].

From the results, it seems that the "too easy - boredom" and "too hard - frustration" combinations do not always apply, because 2 participants reported frustration in the slow-mode and 2 others reported boredom in the fast-mode. Nevertheless, a strong majority (83%) reported boredom in the slow-mode. Although the fast-mode was considered enjoyable by some (58%) and frustrating by others (33%), almost all participants (96%) indicated in the post-game interview to prefer the normal mode over the slow- and fast-mode. Therefore, we conclude that I) the slow-mode was considered boring, II) the fast-mode was enjoyable for some but frustrating for others and III) the normal-mode was the most enjoyable mode. A series of t-tests was performed to investigate the influence of the factor game speed on the emotion-data, as shown in Table 2.

From the preliminary results of this experiment, displayed in Table 2, we can e.g. conclude that the players' mean skin conductance level (SCL) during the slow-mode was significantly different from that during the other two speed-modes (i.e. slow and fast).

## 4 Discussion and Further Work

As argued for, we believe that games can strongly benefit from analyzing the user's affective state when playing a game. The research fields of e.g. affective computing and psycho(physio)logy offer several potentially useful techniques for continuous

analysis of a player's emotional state. The first experiment in the EMO-Pacman project has provided some preliminary empirical evidence for this. The project's ultimate goal is to create an emotionally adaptive game demonstrator. The results from Table 2 can be useful in creating such a classifier. For instance, when a player's skin conductance level drops low during Emo-Pacman, this may indicate boredom because I) the strong majority of subjects in the first experiment found the slow-mode boring, and II) on average (over 24 subjects) the SCL values were significantly lower during this speed mode than during the other two modes. Even though differences in physiological responses between individuals and within individuals over time are not uncommon (see e.g. [19]), the results from Table 2 can be used as a starting point.

One of the main challenges ahead is to create a properly functioning decision making system. For instance, how to respond when two emotion data features indicate boredom but two others do not? In this context, AI-based techniques (e.g. [20]) are expected to be more powerful than only performing statistical tests. Therefore, as a first step in the next phase, a classifier will be tested with the present dataset. In case the results of this test will prove unsatisfactory (e.g. insufficient data/accuracy), unused additional features from the gathered dataset will be analyzed. For instance, the subjects' faces were recorded with a webcam during the experiment (providing possibilities for facial emotion tracking). In addition, other possibilities lie e.g. in analyzing the initial/previous physiological states [21]<sup>2</sup> of the players, analyzing the amplitude / recovery time of their skin conductance responses and looking at the variance, skewness and kurtosis of the emotion-data ([22] and [23] provide more detailed feature analysis descriptions).

## Acknowledgments

The authors would like to thank Gert-Jan de Vries, Jack van den Eerenbeemd, Paul Lemmens and Floris Crompvoets of Philips Research Laboratories and the FUGA project for contributing to this work.

## References

1. Adams, E., Rollings, A.: Game Design and Development; Fundamentals of Game Design. Pearson Education, NJ (2007)
2. Koster, R.: Theory of Fun for Game Design. Paraglyph Press, Phoenix (2004)
3. Pagulayan, R.J., Keeker, K., Wixon, D., Romero, R., Fuller, T.: User-centered design in games. In: Jacko, J., Sears, A. (eds.) Handbook for Human–Computer Interaction in Interactive Systems, pp. 883–906. Lawrence Erlbaum Associates Inc, Mahwah (2002)
4. Bartle, R.A.: Hearts, Clubs, Diamonds, Shades: Players who suit MUDs, <http://www.mud.co.uk/richard/hcds.htm>
5. Öhman, A.: The Psychophysiology of Emotion: An Evolutionary-Cognitive Perspective. In: Ackles, P.K., Jennings, J.R., Coles, M.G.H. (eds.) Advances in Psychophysiology, vol. 2, pp. 79–127. JAI Press, Greenwich (1987)

---

<sup>2</sup> For example, a high (140 BPM) heart rate might not increase as much by an arousing game effect as a normal heart rate.

6. The HUMAINE portal; Research on Emotions and Human-Machine Interaction, <http://emotion-research.net>
7. Ravaja, N.: Contributions of Psychophysiology to Media Research: Review and Recommendations. *Media Psychology* 6, 193–235 (2004)
8. Allanson, J., Fairclough, S.H.: A research agenda for physiological computing. *Interacting with Computers* 16(5), 857–878 (2004)
9. Sundström, P., Ståhl, A., Höök, K.: eMoto - A User-Centred Approach to Affective Interaction. In: Tao, J., Tan, T., Picard, R.W. (eds.) ACII 2005. LNCS, vol. 3784. Springer, Heidelberg (2005)
10. Schaefer, F., Haarmann, B.W.: The Usability of Cardiovascular and Electrodermal Measures for Adaptive Automation. In: Westerink, J.H.D.M., Ouwerkerk, M., Overbeek, T.J.M., Pasveer, W.F., De Ruyter, B. (eds.) Probing Experience: From Assessment of User Emotions and Behavior to Development of Products. Philips Research Book Series, vol. 8, pp. 235–243 (2008)
11. Takahashi, M., Tsuyoshi, A., Kuba, O., Yoshikawa, H.: Experimental Study Toward Mutual Adaptive Interface. In: Proceedings of the 3rd IEEE International Conference on Robot and Human Communication, Nagoya, Japan, pp. 271–276 (1994)
12. Rani, P., Sarkar, N., Liu, C.: Maintaining Optimal Challenge in Computer Games Through Real-Time Physiological Feedback. In: Proceedings of the 1st International Conference on Augmented Cognition, Las Vegas, NV (2005)
13. Journey to Wild Divine, <http://www.wilddivine.com>
14. Bersak, D., McDarby, G., Augenblick, N., McDarby, P., McDonnell, D., McDonald, B., Karkun, R.: Intelligent Biofeedback Using an Immersive Competitive Environment. In: Abowd, G.D., Brumitt, B., Shafer, S. (eds.) UbiComp 2001. LNCS, vol. 2201. Springer, Heidelberg (2001)
15. Saari, T.: Mind-Based Media and Communications Technologies. How the Form of Information Influences Felt Meaning. *Acta Universitatis Tamperensis* 834. Tampere University Press, Tampere (2001)
16. Saari, T., Ravaja, N., Turpeinen, M., Kallinen, K.: Emotional Regulation System for Emotionally Adapted Games. In: Proceedings of FuturePlay 2005, Michigan State University, MI (2005)
17. Tijs, T.J.W., Brokken, D., IJsselsteijn, W.A.: Creating an Emotionally Adaptive Game (submitted)
18. Overmars, M., McQuown, B.: Pac-Man. GM6 [computer software]
19. Lacey, J.I., Lacey, B.C.: Verification and extension of the principle of autonomic response stereotypy. *Am. J. Psychol.* 71, 50–73 (1958)
20. Yannakakis, G.N., Hallam, J.: Towards Optimizing Entertainment in Computer Games. *Applied Artificial Intelligence* 21, 933–971 (2007)
21. Berntson, G.G., Cacioppo, J.T., Quigley, K.S.: Autonomic Determinism: The Modes of Autonomic Control, the Doctrine of Autonomic Space, and the Laws of Autonomic Constraint. *Psychological Review* 98(4), 459–487 (1991)
22. Picard, R., Vyzas, E., Healey, J.: Toward Machine Emotional Intelligence: Analysis of Affective Physiological State. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23(10), 1175–1191 (2001)
23. Van den Broek, E.L., Schut, M.H., Westerink, J.H.D.M., Van Herk, J., Tuinenbreijer, K.: ECCV 2006 Workshop on HCI. In: Huang, T.S., Sebe, N., Lew, M., Pavlović, V., Kölisch, M., Galata, A., Kisačanin, B. (eds.) ECCV 2006 Workshop on HCI. LNCS, vol. 3979, pp. 52–63. Springer, Heidelberg (2006)

# The Game Experience Questionnaire

**Citation for published version (APA):**

IJsselsteijn, W. A., de Kort, Y. A. W., & Poels, K. (2013). *The Game Experience Questionnaire*. Eindhoven: Technische Universiteit Eindhoven.

**Document status and date:**

Published: 01/01/2013

**Document Version:**

Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

**Please check the document version of this publication:**

- A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

[Link to publication](#)

**General rights**

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

[www.tue.nl/taverne](http://www.tue.nl/taverne)

**Take down policy**

If you believe that this document breaches copyright please contact us at:

[openaccess@tue.nl](mailto:openaccess@tue.nl)

providing details and we will investigate your claim.

## **GAME EXPERIENCE QUESTIONNAIRE**

**IJsselsteijn, W.A., de Kort, Y.A.W. & Poels, K.**

## Table of Contents

|    |   |   |
|----|---|---|
| 1. | Introduction.....                                   | 3 |
| 2. | Game Experience Questionnaire – Core Module.....    | 4 |
| 3. | In-game GEQ .....                                   | 6 |
| 4. | GEQ - Social Presence Module.....                   | 7 |
| 5. | GEQ – post-game module.....                         | 8 |
| 6. | Scoring guidelines .....                            | 9 |
|    | Scoring guidelines GEQ Core Module .....            | 9 |
|    | Scoring guidelines GEQ In-Game version .....        | 9 |
|    | Scoring guidelines GEQ Social Presence Module ..... | 9 |
|    | Scoring guidelines GEQ Post-game Module.....        | 9 |

## 1. Introduction

This document contains the English version of the Game Experience Questionnaire. The development and testing of the Game Experience Questionnaire is described in project Deliverable 3.3.

The Game Experience Questionnaire has a modular structure and consists of :

1. The core questionnaire
2. The Social Presence Module
3. The Post-game module.

In addition to these modules, a concise in-game version of the GEQ was developed.

All three modules are meant to be administered immediately after the game-session has finished, in the order given above. Part one and two probe the players' feelings and thoughts while playing the game; Part 3, the post-game module, assesses how players felt after they had stopped playing.

Part 1 is the core part of the GEQ. It assesses game experience as scores on seven components: Immersion, Flow, Competence, Positive and Negative Affect, Tension, and Challenge. For a robust measure, we need five items per component. As translation of questionnaire items, no matter how carefully performed, sometimes results in suboptimal scoring patterns, we have added a spare item to all components. After the first use of the translated GEQs, scale analyses will be performed to check whether any item should be discarded or replaced.

Part 2, the social presence module, investigates psychological and behavioural involvement of the player with other social entities, be they virtual (i.e., in-game characters), mediated (e.g., others playing online), or co-located. This module should only be administered when at least one of these types of co-players were involved in the game.

Part 3, the post-game module, assesses how players felt after they had stopped playing. This is a relevant module for assessing naturalistic gaming (i.e., when gamers have voluntarily decided to play), but may also be relevant in experimental research.

The In-game version of the GEQ is a concise version of the core questionnaire. It has an identical component structure and consists of items selected from this module. The in-game questionnaire is developed for assessing game experience at multiple intervals during a game session, or play-back session. This should facilitate the validation of continuous and real-time indicators some of the partners in the FUGA project are developing.

## 2. Game Experience Questionnaire – Core Module

Please indicate how you felt while playing the game for each of the items,  
on the following scale:

| not at all | slightly | moderately | fairly | extremely |
|------------|----------|------------|--------|-----------|
| 0          | 1        | 2          | 3      | 4         |
| < >        | < >      | < >        | < >    | < >       |

- 1 I felt content
- 2 I felt skilful
- 3 I was interested in the game's story
- 4 I thought it was fun
- 5 I was fully occupied with the game
- 6 I felt happy
- 7 It gave me a bad mood
- 8 I thought about other things
- 9 I found it tiresome
- 10 I felt competent
- 11 I thought it was hard
- 12 It was aesthetically pleasing
- 13 I forgot everything around me
- 14 I felt good
- 15 I was good at it
- 16 I felt bored
- 17 I felt successful
- 18 I felt imaginative
- 19 I felt that I could explore things
- 20 I enjoyed it
- 21 I was fast at reaching the game's targets
- 22 I felt annoyed
- 23 I felt pressured
- 24 I felt irritable
- 25 I lost track of time
- 26 I felt challenged
- 27 I found it impressive
- 28 I was deeply concentrated in the game
- 29 I felt frustrated
- 30 It felt like a rich experience
- 31 I lost connection with the outside world
- 32 I felt time pressure

33 I had to put a lot of effort into it

### 3. In-game GEQ

Please indicate how you felt while playing the game for each of the items, on the following scale:

|            |          |            |        |           |
|------------|----------|------------|--------|-----------|
| not at all | slightly | moderately | fairly | extremely |
| 0          | 1        | 2          | 3      | 4         |
| < >        | < >      | < >        | < >    | < >       |

- |   |               |
|---|---------------|
| 1 I was interested in the game's story  | GEQ Core – 3  |
| 2 I felt successful                     | GEQ Core – 17 |
| 3 I felt bored                          | GEQ Core – 16 |
| 4 I found it impressive                 | GEQ Core – 27 |
| 5 I forgot everything around me         | GEQ Core – 13 |
| 6 I felt frustrated                     | GEQ Core – 29 |
| 7 I found it tiresome                   | GEQ Core – 9  |
| 8 I felt irritable                      | GEQ Core – 24 |
| 9 I felt skilful                        | GEQ Core – 2  |
| 10 I felt completely absorbed           | GEQ Core – 5  |
| 11 I felt content                       | GEQ Core – 1  |
| 12 I felt challenged                    | GEQ Core – 26 |
| 13 I had to put a lot of effort into it | GEQ Core – 33 |
| 14 I felt good                          | GEQ Core – 14 |

#### 4. GEQ - Social Presence Module

Please indicate how you felt while playing the game for each of the items, on the following scale:

| not at all | slightly | moderately | fairly | extremely |
|------------|----------|------------|--------|-----------|
| 0          | 1        | 2          | 3      | 4         |
| < >        | < >      | < >        | < >    | < >       |

- 1 I empathized with the other(s)
- 2 My actions depended on the other(s) actions
- 3 The other's actions were dependent on my actions
- 4 I felt connected to the other(s)
- 5 The other(s) paid close attention to me
- 6 I paid close attention to the other(s)
- 7 I felt jealous about the other(s)
- 8 I found it enjoyable to be with the other(s)
- 9 When I was happy, the other(s) was(were) happy
- 10 When the other(s) was(were) happy, I was happy
- 11 I influenced the mood of the other(s)
- 12 I was influenced by the other(s) moods
- 13 I admired the other(s)
- 14 What the other(s) did affected what I did
- 15 What I did affected what the other(s) did
- 16 I felt revengeful
- 17 I felt schadenfreude (malicious delight)

## 5. GEQ – post-game module

Please indicate how you felt after you finished playing the game for each of the items, on the following scale:

|            |          |            |        |           |
|------------|----------|------------|--------|-----------|
| not at all | slightly | moderately | fairly | Extremely |
| 0          | 1        | 2          | 3      | 4         |
| < >        | < >      | < >        | < >    | < >       |

- 1 I felt revived
- 2 I felt bad
- 3 I found it hard to get back to reality
- 4 I felt guilty
- 5 It felt like a victory
- 6 I found it a waste of time
- 7 I felt energised
- 8 I felt satisfied
- 9 I felt disoriented
- 10 I felt exhausted
- 11 I felt that I could have done more useful things
- 12 I felt powerful
- 13 I felt weary
- 14 I felt regret
- 15 I felt ashamed
- 16 I felt proud
- 17 I had a sense that I had returned from a journey

## 6. Scoring guidelines

### Scoring guidelines GEQ Core Module

The Core GEQ Module consists of seven components; the items for each are listed below.

Component scores are computed as the average value of its items.

**Competence:** Items 2, 10, 15, 17, and 21.

**Sensory and Imaginative Immersion:** Items 3, 12, 18, 19, 27, and 30.

**Flow:** Items 5, 13, 25, 28, and 31.

**Tension/Annoyance:** Items 22, 24, and 29.

**Challenge:** Items 11, 23, 26, 32, and 33.

**Negative affect:** Items 7, 8, 9, and 16.

**Positive affect:** Items 1, 4, 6, 14, and 20.

### Scoring guidelines GEQ In-Game version

The In-game Module consists of seven components, identical to the core Module. However, only two items are used for every component. The items for each are listed below.

Component scores are computed as the average value of its items.

**Competence:** Items 2 and 9.

**Sensory and Imaginative Immersion:** Items 1 and 4.

**Flow:** Items 5 and 10.

**Tension:** Items 6 and 8.

**Challenge:** Items 12 and 13.

**Negative affect:** Items 3 and 7.

**Positive affect:** Items 11 and 14.

### Scoring guidelines GEQ Social Presence Module

The Social Presence Module consists of three components; the items for each are listed below.

Component scores are computed as the average value of its items.

**Psychological Involvement – Empathy:** Items 1, 4, 8, 9, 10, and 13.

**Psychological Involvement – Negative Feelings:** Items 7, 11, 12, 16, and 17.

**Behavioural Involvement:** Items 2, 3, 5, 6, 14, and 15.

### Scoring guidelines GEQ Post-game Module

The post-game Module consists of four components; the items for each are listed below.

Component scores are computed as the average value of its items.

**Positive Experience:** Items 1, 5, 7, 8, 12, 16.

**Negative experience:** Items 2, 4, 6, 11, 14, 15.

**Tiredness:** Items 10, 13.

**Returning to Reality:** Items 3, 9, and 17.

## D3.3 : Game Experience Questionnaire

### **Citation for published version (APA):**

Poels, K., de Kort, Y. A. W., & IJsselsteijn, W. A. (2007). *D3.3 : Game Experience Questionnaire: development of a self-report measure to assess the psychological impact of digital games*. Eindhoven: Technische Universiteit Eindhoven.

### **Document status and date:**

Published: 01/01/2007

### **Document Version:**

Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

### **Please check the document version of this publication:**

- A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

[Link to publication](#)

### **General rights**

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

[www.tue.nl/taverne](http://www.tue.nl/taverne)

### **Take down policy**

If you believe that this document breaches copyright please contact us at:

[openaccess@tue.nl](mailto:openaccess@tue.nl)

providing details and we will investigate your claim.



**028765**  
**FUGA**  
**The fun of gaming:**  
**Measuring the human experience of media enjoyment**

**STREP / NEST-PATH**  
**Deliverable D3.3:**  
**GAME EXPERIENCE QUESTIONNAIRE**

**K. Poels, Y.A.W. de Kort & W.A. IJsselsteijn**

**TU/e**

|                                  |  |
|----------------------------------|--|
| <b>Title of Contract</b>         | <b>FUGA - The fun of gaming: Measuring the human experience of media enjoyment</b>   |
| <b>Acronym</b>                   | <b>FUGA</b>  |
| <b>Contract Number</b>           | <b>NEST-PATH-028765</b>  |
| <b>Start date of the project</b> | <b>01.05.2006</b>  |
| <b>Duration</b>                  | <b>36 months, until 30.4.2009</b>  |
| <b>Reporting period</b>          |  |
| <b>Date of preparation</b>       |  |
| <b>Project coordinator</b>       | <b>Name: Niklas Ravaja<br/>Organization: Helsinki School of Economics<br/>Phone: +358 40 411 4565<br/>Fax: +358 9 431 38391<br/>E-Mail: <a href="mailto:niklas.ravaja@hse.fi">niklas.ravaja@hse.fi</a><br/>Project web site: <a href="http://www.hse.fi/fuga">http://www.hse.fi/fuga</a></b> |

**Partners:**

|                  |    |
|------------------|----|
| Partner 1 (CKIR) | FI |
| Partner 2 (TKK)  | FI |
| Partner 3 (HGO)  | SE |
| Partner 4 (HMTH) | DE |
| Partner 5 (UKA)  | DE |
| Partner 6 (TUE)  | NL |



Project funded by the European Community under the FP6  
New and Emerging Science and Technology (NEST) programme

## Executive summary

The current report describes the development of the Game Experience Questionnaire, a self-report measure that aims to comprehensively and reliably characterise the multifaceted experience of playing digital games. Its development and evaluation precede the testing of the other measures of game experience targeted in the FUGA project, as it will be employed for evaluating validity of these measures.

The theoretical work performed by all partners in the FUGA project (Deliverable 2.1) provided a firm and broad basis for the development of the scale. In addition, relevant scientific literature was reviewed for any conceptualisations that might complement this work and for existing scales that might serve as a starting point or inspiration for item formulation. Furthermore, empirical data was gathered during focus group interviews with different types of gamers. These interviews served as a test for comparing scientific conceptualisations and lay descriptions of first-hand experiences. They were also used as a reference guide on choice of wording in the item formulation phase.

The Game experience questionnaire is developed with a modular structure, consisting of:

1. The core questionnaire (GEQ). This is the heart of the Game Experience Questionnaire, probing multiple components of players' experience while gaming.
2. The post-game questionnaire (PGQ), probing gamers' experience after the gaming session and any after effects.
3. The social presence module (SPGQ), probing gamers' experience of and involvement with their co-player(s).

These three lists are to be administered after the gaming session has ended. Additionally, a short in-game version of the GEQ was developed, the iGEQ, for probing in-game experience multiple times during a gaming session.

A large scale survey was performed to test the long list (>100 items) and explore the factor structure of the questionnaires. Factor analyses provided a structure for the scale that made good sense in the light of theoretical considerations, with subscales that were all easy to interpret. Subsequent reliability tests resulted in the construction of robust subscales with satisfactory to high internal consistencies.

Additional explorations were subsequently performed to check sensitivity and validity of the developed measures. Statistical analyses firmly demonstrated that the GEQ and additional modules were sensitive enough to pick up differences between gamers, game types, play characteristics, and social context of play. The findings already provide new insights to scholars in the field. The Game Experience Questionnaire is now ready to be translated and used in FUGA experimentations.

## Table of Contents

|  |           |
|--|-----------|
| <b>1. INTRODUCTION .....</b>   | <b>4</b>  |
| 1.1. CRITERIA FOR GAME EXPERIENCE MEASURES .....                                       | 5         |
| <b>2. EXPLORATIONS OF DIGITAL GAME EXPERIENCE: FOCUS GROUPS AND EXPERT REVIEW.....</b> | <b>7</b>  |
| 2.1. FOCUS GROUP METHODOLOGY.....  | 7         |
| 2.2. FOCUS GROUPS ABOUT DIGITAL GAME EXPERIENCE .....                                  | 7         |
| 2.2.1. <i>Method</i> .....   | 8         |
| 2.2.2. <i>Results</i> .....  | 9         |
| 2.2.3. <i>Conclusions</i> .....  | 13        |
| 2.3. EXPERT MEETING .....  | 13        |
| <b>3. DEVELOPMENT OF THE GAME EXPERIENCE QUESTIONNAIRE .....</b>                       | <b>16</b> |
| 3.1. GENERAL LOGIC OF THE GEQ DEVELOPMENT: CORE AND MODULAR ELEMENTS.....              | 16        |
| 3.2. ITEM GENERATION .....   | 16        |
| 3.3. RESEARCH METHOD.....  | 17        |
| 3.4. EXPLORATORY FACTOR ANALYSIS AND SCALE RELIABILITY.....                            | 18        |
| 3.4.1. <i>Rationale for statistical procedure</i> .....                                | 18        |
| 3.4.2. <i>GEQ – core</i> .....   | 19        |
| 3.4.3. <i>In-game questionnaire (iGEQ)</i> .....                                       | 22        |
| 3.4.4. <i>Post-game questionnaire (PGQ)</i> .....                                      | 22        |
| 3.4.5. <i>GEQ – social presence module</i> .....                                       | 23        |
| 3.5. CONCLUSION .....  | 25        |
| <b>4. EXPLORATIONS OF THE GEQ: INTER-CORRELATIONS AND BASIC SENSITIVITY .....</b>      | <b>26</b> |
| 4.1. CORE GAME EXPERIENCE QUESTIONNAIRE (GEQ) .....                                    | 26        |
| 4.2. IN-GAME QUESTIONNAIRE (iGEQ) .....  | 30        |
| 4.3. POST-GAME QUESTIONNAIRE (PGQ).....  | 31        |
| 4.4. SOCIAL PRESENCE IN GAMING QUESTIONNAIRE (SPGQ).....                               | 34        |
| 4.5. CONCLUSION .....  | 36        |
| <b>5. DISCUSSION AND CONCLUSION .....</b>  | <b>38</b> |
| 5.1. QUESTIONNAIRE DEVELOPMENT .....   | 38        |
| 5.2. FURTHER STEPS IN THE DEVELOPMENT OF THE GEQ .....                                 | 39        |
| <b>REFERENCES.....</b>   | <b>40</b> |
| <b>APPENDIX 1: GAME EXPERIENCE QUESTIONNAIRE.....</b>                                  | <b>41</b> |
| <b>APPENDIX 2: IN-GAME GEQ.....</b>  | <b>43</b> |
| <b>APPENDIX 3: POST-GAME EXPERIENCE QUESTIONNAIRE (PGQ) .....</b>                      | <b>44</b> |
| <b>APPENDIX 4: SOCIAL PRESENCE GAMING QUESTIONNAIRE (SPGQ).....</b>                    | <b>45</b> |
| <b>APPENDIX 5: SCORING GUIDELINES .....</b>  | <b>46</b> |
| Scoring guidelines GEQ Core Module .....   | 46        |
| Scoring guidelines GEQ In-Game version.....  | 46        |
| Scoring guidelines GEQ Social Presence Module.....                                     | 46        |
| Scoring guidelines GEQ Post-game Module .....  | 46        |

## 1. Introduction

Over recent years, digital games have rapidly gained in interest. In pop culture they rival film and television in popularity. Originally deemed frivolous, game design principles and game engines are now being re-used for purposes of education, training and therapy, spawning an area now known as 'serious games'. Meanwhile, digital games have significantly diversified in design. From classic arcade games to embodied games (e.g., the Nintendo Wii), mixed reality games, and pervasive games. Research on digital games has been thematically and disciplinary diverse, perhaps born from the multifaceted nature of digital games themselves. Scholars from a variety of perspectives, including computer science, visual design, film and television theory, performing arts, literary theory, and media psychology, to name but a few, have taken an interest in digital games. More recently, HCI as an academic discipline has turned its attention towards digital games, finding inspiration for engaging interface design, creative new metaphors for human-computer interaction, and generally focusing on the captivating experiences that can be designed through this class of computer-based technologies.

Today, the area of digital games constitutes a tremendously varied set of applications, with a wide range of associated player experiences, defying a one-size-fits-all approach to its conceptualisation and measurement. One of the main challenges facing the gaming research community is a lack of a coherent and fine-grained set of methods and tools that enable the measurement of entertainment experiences in a sensitive, reliable and valid manner. Much like the six wise (but blind) men touching the elephant, no single methodological perspective can be said to provide a comprehensive understanding of digital gaming. Following this insight, the FUGA consortium takes a critical view towards the exclusive reliance on any one single indicator for measuring game experience. In the FUGA project, we explicitly strive towards a multi-method, multi-measure approach whereby we anchor and cross-validate various measures (e.g., self-report, psychophysiological, behavioural, neural) via their simultaneous application to a certain standardized set of games, and correlating the results thus obtained. Assessing the psychometric properties (sensitivity, reliability, validity) of all measures developed in the project is one of the defining characteristics of FUGA.

We believe that a large range of measures, from reflective (subjectively controllable) to fully reflexive (uncontrollable) responses, enables a fuller characterization of the game experience than any single isolated measure, thus sensitizing us to the rich gamut of experiences associated with digital games. Moreover, limitations particular to one measure may be overcome or compensated by using corroborating evidence emerging from another measure. The combination of multiple measurement modalities can thus reduce uncertainty associated with measuring a single modality, resulting in increased validity, robustness and wider applicability of the total set of measures.

As a significant first step, we report in this deliverable on the development and validation of the Game Experience Questionnaire (GEQ). The GEQ is intended to become a freely available and widely applicable measure which allows researchers to obtain a reliable and valid indication of participants' *subjective experiences* associated with digital gameplay, both during and after a gaming episode. The GEQ covers a wide range of digital game experiences that have been identified through reviewing theoretical accounts of player experiences (for an overview see FUGA Deliverable 2.1 'Working Model of Digital Game Experience'), as well as through focus group explorations with both frequent and infrequent gamers. The factor structure of the GEQ, which has been established based on a large-scale survey, is very much in line with the theoretical constructs that have been identified in the FUGA Working Model of Digital Game Experience.

The current deliverable documents the development and evaluation of the Game Experience Questionnaire. In the remainder of this chapter, we will briefly discuss a number of relevant psychometric quality criteria that any game experience measure needs to adhere to. In addition, we will highlight some of the quality criteria that are of particular relevance in questionnaire design. Chapter 2 describes in detail the results of a number of focus group explorations of digital game experience. These focus groups provided us with important insights into lay conceptualisations of game experience, and were a valuable addition to the theoretical groundwork that was reported on in Deliverable 2.1. In an expert meeting both theoretical perspectives and focus group results were subsequently combined, providing a solid foundation for the item generation phase of the questionnaire development. Chapter 3 describes the development of the GEQ, including the general logic of the GEQ, item generation and selection, scale construction based on exploratory factor analysis, and internal consistency. Chapter 4 further explores the sensitivity of the GEQ scales to differentiate between gamers, game types, play characteristics, and variations in the social context of

play. Finally, Chapter 5 provides a brief summary and future outlook, describing the translation procedure of the GEQ, open issues, and plans for future validation.

## 1.1. Criteria for game experience measures

Several criteria or properties can be formulated to which a good measure of game experience should adhere. These are reliability, validity, sensitivity, robustness, non-intrusiveness, and convenience. From a psychometrical point of view, reliability, validity, and sensitivity are the three most important criteria that should be satisfied. More information on basic psychometrical properties of tests can be found in Cronbach (1990). We will briefly summarise the main criteria below.

### Reliability

A measure should be consistent and stable over time, meaning that it should give comparable results if administered under comparable conditions. There are several ways to estimate reliability:

- Internal consistency reliability: the extent to which the items of a measure address the same underlying trait or characteristic. There are different ways of estimating internal consistency: Cronbach's alpha, (average) inter-item correlation, (average) item-total correlation, and split-half reliability.
- Test-retest reliability: the stability of a measure over time. Test-retest reliability can be calculated as the correlation between scores gathered at different occasions. Researchers should make sure that it can be reasonably assumed that the measured construct itself has not changed between these occasions.
- Inter-rater reliability: the degree to which different observers agree in their assessment. Inter-rater reliability can be calculated as the percentage of cases in which the observers give the same rating (for nominal measurement) or the correlation between ratings (for ordinal, interval, or ratio measurement)
- Parallel-forms: the consistency of similar measures. Parallel forms can be created by generating a large set of items addressing the same concept and randomly dividing them into two sets, administering them, and calculating the correlation. A related approach is to divide an existing measurement tool into two halves randomly, and calculate the correlation between the two scores. This is called split-half reliability.

Ideally, reliability should be assessed in several ways in order to draw firm conclusions about a measurement instrument. In reality, however, this is often not possible. Internal consistency is the most widely used form of reliability, because it can be assessed using only one measurement instrument, on one occasion, in one population. For the Game Experience Questionnaire developed in this project, we report on the internal consistency (Cronbach's alpha) of all subscales and the total questionnaire in the current deliverable (Chapter 3). The test-retest reliability of the entire battery of measures developed by FUGA (including the GEQ) will be established in WP6. Inter-rater reliability is only of relevance in cases where multiple raters are employed to score observational data. In FUGA, this is of particular relevance in the case of behavioural measures, where video observations need to be coded into uniform categories of behaviour by multiple observers.

### Validity

A measure should address the intended construct. There are many different approaches to validity, the most important of which are discussed here.

- Face validity: the extent to which a measure appears to address the intended construct. This approach is based on subjective judgment, preferably of several experts.
- Content validity: the instrument is checked against the relevant content domain for the construct, to see whether the instrument is compatible with theories and addresses all relevant dimensions of the construct. For this approach, it is necessary that theories about and clear definitions of the construct and its dimensions exist.
- Criterion-related validity: comparing the measure to some other measure or criterion. Different forms of criterion-related validity are predictive validity (the extent to which a theoretically relevant criterion can be predicted), convergent validity (the degree to which the measure correlates with measures of theoretically related constructs), and discriminant

validity (the degree to which a measure is different from measures of constructs to which it is not theoretically similar).

Face validity and content validity are typically established in the construction phase of a measurement instrument. For the GEQ, they are discussed in Chapters 2 and 3. Criterion-related validity can only be established through research. It is a more objective and therefore more convincing indication of validity. Predictive validity will be established through a series of studies performed under WP7. Convergent validity is in fact at the heart of the FUGA multi-measure approach, as we are comparing and correlating the outcomes of different measurement methods throughout the project lifetime. For the GEQ, discriminant validity is established internally for the various subscales, as one would expect different scales to behave differently depending on the particulars of the game content, interface and setting.

### **Sensitivity**

A measure should be able to distinguish between different levels of a construct with a reasonable level of detail. For example, if different levels of (a component of) game experience are expected based on different form factors, different game content or different individual characteristics, the measure should reflect this difference. For FUGA, temporal sensitivity of a measure is also of particular importance, since the range of experiences we are interested in will likely vary over time within the same gaming session. In Chapter 4 of the present deliverable, we analyse the sensitivity of the GEQ in relation to gender, play frequency, type of game, and social setting.

### **Robustness**

A measure should be applicable across a variety of different media platforms and settings, varying in form, content, and context-of-use. The GEQ has explicitly been developed to be a widely applicable tool, offering a valid and reliable experiential assessment across gaming genres, platforms, play styles, social settings, etc.

### **Non-intrusiveness**

A measure should be as non-intrusive and non-disruptive as possible, and not interfere with the construct that is being measured. For example, when the measurement method requires self-report during the game, this may interfere with game experiences that build on high levels of mental absorption or attentional engagement (e.g., flow, immersion). Measurement devices that are uncomfortable, restrict movement, or that are otherwise highly noticeable may also affect the experience under study. Most real-time measures will inevitably be intrusive to a certain extent (e.g., psychophysiology, fMRI). The GEQ, being a post-test self-report measure, can be assumed to be non-intrusive.

### **Convenience**

A measure should preferably be easy to learn, easy to administer, low-cost, and portable. This will make it much easier to apply a measure outside the confines of a psychological lab, potentially adding to the measure's ecological validity. As a low-cost paper-based measure, the GEQ can be regarded as highly convenient for researchers to use.

## 2. Explorations of digital game experience: focus groups and expert review

In this chapter we describe the exploratory part of our conceptualization of game experience dimensions. We executed this exploratory part in two stages. First, focus groups were organised to obtain lay-conceptualisations of game experience and second, findings were discussed in an expert meeting with the aim of consolidating empirical and theoretical findings into a tentative model of game experience. We describe those two stages below.

### 2.1. Focus group methodology

Focus group methodology is a qualitative research tool that is frequently used in social sciences to explore people's meanings, ways of understanding, or experiences of a complex phenomenon (Lunt & Livingstone, 1996). In practice, focus group methodology typically involves a series of group interviews about a given topic or phenomenon guided by a moderator. The moderator plays a pivotal role. Not only does he or she put forward the questions and concrete topics that need to be discussed, he or she also ensures that the discussion remains on the topic at hand (Lunt & Livingstone, 1996). Focus group sizes usually vary from five to eight participants (Morgan, 1998). A rule of thumb with respect to the amount of focus groups is that one should continue to run new groups until the last groups has nothing fundamentally new to add (Lunt & Livingstone, 1996).

One of the major strengths of focus group methodology is its *exploratory nature*. Focus groups enable the researcher to get to know their target audience in detail without the need for a priori assumptions or research questions. Moreover, focus groups can serve as a source of new ideas and hypotheses (Merton, 1987). Further, focus groups are very useful in providing *context and depth*. Besides observing experiences and thoughts, the moderator can probe in order to acquire relevant background information (e.g., about motivations, contexts) on these experiences and thoughts. Related to this, focus group methodology lends itself for *interpretation* of the experiences and thoughts reported by the target audience. As such, it enables researchers to get a clearer view on the *why* of behaviour (Morgan, 1998).

### 2.2. Focus groups about digital game experience

Focus group methodology is an interesting research tool to explore several facets of digital gaming. For example, Miller et al. (1996) applied focus groups to explore female preferences for specific types of game designs. Their research findings report that girls placed a premium on richly textured video and audio. Girls also preferred collaborating to competing; they liked interacting with male characters, and showed a preference for simulation games. Applying in-depth interviews (i.e., a related qualitative research tool in which the participants are interviewed individually) Brown and Cairns (2004) unravelled some core aspects of the concept of game immersion.

Given the diversity of individual differences with respect to play styles (Bartle, 1996) or motivations to play games (Yee, 2002), focus groups can provide in depth, contextual, and motivational insights into the specific experiences of different types of gamers. To the best of our knowledge, the application of focus group methodology to study digital game experience is still limited.

In the FUGA project we use focus group methodology to *explore lay meanings and experiences of digital gaming*. The underlying reasons are twofold. First, in order to develop a long list of items as a starting point for our Game Experience Questionnaire, we use focus groups as one input source for item generation (next to theory – see D2.1 and expert insights – see section 4.3 of this document). Lay verbalizations can provide a rich perspective on possible game experiences. Moreover, some of them might have been overlooked by current theoretical conceptualizations. Further, lay verbalizations do also aid us in choosing the appropriate formulations of the different game experiences and, as such, ensure face validity of the different items that will be used in the Game Experience Questionnaire. An additional reason to use focus group methodology is to get more insights in the game experiences of different types of gamer like frequent gamers versus infrequent gamers.

### 2.2.1. Method

We organized three focus groups. Two focus groups (FG1 and FG2) included infrequent gamers (i.e., people who game at least once a month). The third focus group (FG3) consisted of frequent gamers (i.e., people who game at least once a week). Participants were undergraduates and graduated students. Their ages ranged from 19 to 28 years. Specifically, FG1 had five participants of which two were female. FG2 consisted of three male participants. FG3 had four male participants. Each focus group took about 90 minutes and participants were rewarded 10 € for their participation.

The focus groups were structured in the following way:

*Introductory round:* First, the moderator and the assistant moderator presented themselves and gave a brief description of the main goal of the focus groups. More concretely, they explained that the focus group was about game experience and participants could freely talk about how they experienced digital gaming. Then, participants presented themselves by briefly introducing themselves, giving their name, game frequency, and the type of games they usually played.

*Individual task:* We asked each participant to reflect for five minutes on what they considered as the most prominent game experiences for themselves. Participants had to write down these experiences on Post-It notes. We also asked them to indicate their most favourite game and the game they had played last. After this, all Post-It's were pasted in the middle of the table to serve as a starting point and inspiration source for the next and most crucial stage, the group discussion (see Picture 1).



*Picture 1. Set up for the group discussion*

*Group discussion:* In the group discussion participants could freely talk and interact with each other about their game experiences. The discussion was clustered around three core questions by means of a semi-structured questionnaire. Accordingly, the three core question were fixed but other, side questions could be posed along the way in order to have clarification or in-depth insights. The three core questions were: (1) On what occasions do you typically start gaming?, (2) What do you experience or feel *while* gaming?, (3) What do you experience or how do you feel *after* gaming? The moderator further probed the experiences that were reported by each participant individually. Additional Post-It's were used when new experiences were mentioned.

*Group task:* At the end of the group discussion participants were asked to cluster and rank all game experiences that were reported on the Post It notes depending on how central they are to gaming in general (i.e., across games). They wrote down all experiences on a large sheet of paper with the

most prominent experiences in the centre of the sheet and the less relevant experiences closer to the margins of the sheet (see Picture 2).

All focus group interviews were recorded and transcribed.



*Picture 2. Group task: ranking the Post-Its on a large sheet of paper.*

### **2.2.2. Results**

In this results section we focus on the three core questions that were posed in the group discussion. We describe results for occasional and frequent gamers separately.

#### **a) Infrequent gamers**

##### ***Question 1: On what occasions do you typically start gaming?***

The occasions in which the infrequent gamers typically start gaming vary considerably. A substantial amount of them reported that they started playing a game when they felt bored, when they had nothing else to do, or when they didn't feel like doing something else (e.g., studying).

*....when I am feeling bored or when I don't feel like studying... (Female participant, FG1, 21 years)*

*I game when I feel like gaming, when I don't feel like doing anything else... (Male participant, FG2, 24 years)*

Related to this, some of the infrequent gamers said that they typically started gaming upon coming home after a busy day at school or after an intensive study session.

*If I come home after a busy day and I don't want to do anything else yet, I often play a couple of quick games before I continue with something else (male participant, FG1, 20 years)*

*I sometimes play a game if I want to relax, de-stress, or divert my thoughts... (Male participant, FG2, 28 years)*

Another occasion that they put forward was more social in nature. Some of the infrequent gamers reported that they often played games when they were with friends, for example, before going out.

*I rarely game on my own. When I game it is a social event where we sit on the couch, with beer and chips. This usually happens the hours before we go out. (Male participant, FG2, 22 years)*

One female participant reported a combination of the social and the boredom occasion.

*When we are together with friends and we have a break or when we do not really know what to do, we sometimes play a game together. (Female participant, FG1, 21 years)*

### **Question 2: What do you experience or feel while gaming?**

Almost all participants mentioned *fun*, *amusement*, and *relaxation* as most prominent game experiences.

Experiences during game play were often related to game immersion and flow. This means, the infrequent gamers mentioned experiences like '*loosing connection with the outside world*', '*forgetting everything around you*', and '*being fully occupied with the game*'. One participant explicitly linked fun with the experience of game immersion.

*Feeling happy is linked with loosing connection with the outside world. It is a simple game and you get yourself fully drawn in. (male participant, FG1, 22 years; talking about playing Mario Bros with his Nintendo console)*

Other experiences were more closely linked to imaginative and sensory immersion. For example, '*being creative*', '*exploring the game world*'. One of the participants linked these experiences to the fun factor:

*I like it when you get more creative in a game. It is funny when you discover something new, something you did not expect. When you find out something that you were looking for, you feel glad. (Female participant, FG1, 21 years)*

Others said that they did not want to invest much time in exploring a game. For them, relaxation was related to simple and clear games.

*I don't game very often so if I do I want to know directly how the game works. I don't want to invest much time in exploring the game. (Male participant, FG2, 22 years)*

Further, '*concentration*' and '*tension*' were mentioned as in-game experiences. Participants reported that these experiences were related to challenge and difficulty of the game. Some of them reported that concentration was needed in order to perform well in games, and, as the game got more difficult, more concentration was needed. However, most of them said that they typically gave up when it got too difficult.

*If it gets difficult, you need to concentrate more. Sometimes I give up, when it really doesn't work. (Male participant, FG1, 19 years)*

Further, a lot of game experiences were connected to playing against co-located others, or playing against the computer. Experiences that are typically mentioned in this context are '*competition*' and '*enjoyment with others*'.

Participants mentioned that competition instigated feelings of '*tension*', '*nervousness*', and '*teasing one another*', while at the same time, they perceived competition as '*fun*', '*having a laugh*', and '*being connected with others*'.

*It is always a lot of fun! For example when we play Mario Cart with four friends, there's a lot of friendly banter. It is very funny if one player gets picked on by three others. That enhances the enjoyment you have with others. (Male participant, FG2, 22 years)*

When probing what happens when they themselves were the 'victim' of this group teasing, the answer was:

*Then I laugh about it, but of course you also feel a little bad. But they often want to get me because I am the one who tries hardest to frustrate them. So, it is always very funny... (Male participant, FG2, 22 years)*

Emotions evoked through competition with co-located people were reported as much stronger than emotions through competition with the computer, or through competition with online people. Also, participants reported that they put more effort in the game when they play against co-located friends.

Moreover, they said that they experienced more tendencies to take 'revenge'. This was attributed to the physical presence, which enables non-verbal and verbal communication and physical contact.

*Playing against the computer is totally different from playing against a friend who sits next to you. You can nudge him, give comments... (Male participant, FG2, 28 years)*

*When you play with strangers on the Internet, you miss a part of the communication. You cannot figure out whether they play for fun or not. You cannot tease them. (Male participant, FG2, 24 years)*

### **Question 3: What do you experience after gaming?**

With regard to experiences after gaming, results were mixed. Most of the participants said that '*time had gone by faster than expected*'. When probing on whether this led to feelings of *regret* or *satisfaction*, the answers varied according to personal and situational factors.

*I often feel bad if I wasted my time with playing a game. However, if it is a lazy Saturday afternoon and you have nothing better to do it doesn't matter. Then I even find it useful to play a game. (Female participant, FG1, 21 years)*

Others also reported feeling bad after gaming longer than they had intended to, especially when they had other, more important things to do.

*Sometimes when you are studying and you take a short break, you forget the time and then you feel bad that you have wasted your precious time at playing stupid games. (Female participant, FG1, 21 years)*

*If the weather is nice, then I regret that I didn't spend my time outside. I find it a pity then. However, I did have fun, so it is not that bad after all. (Male participant, FG2, 28 years)*

Some of them reported that they anticipated those negative experiences. For example, one participant explicitly stated that he only quit gaming when he was in a favourable position. This way, he reported, he always has a good feeling after gaming. Another participant said that he would not start gaming when he had more urgent things to do (e.g., when he had examinations). Yet another mentioned only playing short games in order to prevent that he would spend his whole evening on playing a game.

### **b) Frequent gamers**

#### **Question 1: On what occasions do you typically start gaming?**

There were several different occasions on which the frequent gamers start gaming. On the one hand, they reported that they often started gaming when they had nothing else to do, when looking for relaxation, or when they wanted to divert their thoughts to something else than work or school. This is very similar to the instances mentioned by infrequent gamers. On the other hand, they reported that they played games in a coordinated way, making appointments with friends, and competing with them in a team. In both cases, they played with friends, mostly online. The type of game they played differed with the occasion.

*I game if I want to do something completely different, for example if I come home after work. Most of the time I play a couple of short First Person Shooter (FPS) games, those games you can play at any moment, against anyone. In the evenings, I play longer Real Time Strategy (RTS) games. (Male participant, FG3, 28 years)*

*I play FPS games if I have nothing else to do, or World Worms Party. When I play Massive Multi-player Online Role Playing Games (MMORPG) it happens in a much more coordinated way, you really need to make appointments beforehand. (Male participant, FG3, 23 years)*

#### **Question 2: What do you experience or feel while gaming?**

Frequent gamers found it hard to report their experiences during gaming, since these depended on several factors. In general, their descriptions were richer than those of most infrequent gamers, i.e., involved more facets, e.g., competition, and control.

First, when they reported on their experiences while gaming, they often distinguished between different types of games.

*FPS games are about beating the opponent and are very demanding. As such, the atmosphere and graphics are less important. With MMORPG it is all about the atmosphere and the beautiful scenes. You get yourself fully drawn in.* (male participant, FG3, 29 years)

Second, a lot of the experiences they reported involved game settings in which they cooperate in a team. As such, the social experience is very important. Experiences that emerge from this social setting are 'having fun', 'being powerful', 'having control', 'immersion', 'thrill' and 'satisfaction'.

*It is nice to play in a team; we often make a lot of jokes and fun together. The urge to build something evokes pleasure. The feeling of getting more and more power and more control on your environment is also part of the fun; and also that you get more status within your environment.* (Male participant, FG3, 29 years)

*It is important that you feel that you are one team. For me realism is important so you can fully imagine yourself in the game. It causes more of a thrill. If I experience that I am really someone in the game and for my team, it gives me a feeling of satisfaction.* (Male participant, FG3, 26 years)

Third, they explicitly distinguished between experiences while playing games purely for fun and experiences while playing games in a serious game competition or online in a team.

Particularly, 'immersion', 'concentration', and 'competition' seemed to differ between those two types of game play. Interestingly, frequent gamers reported to play console games when playing for fun and PC games when competing in more serious game competitions.

*With MMORPG and FPS you need to sit close to the screen, they are very exacting. If you meet with friends to have some fun together, it's much nicer to lean back on the couch, the game triggers the fun, but it's also about other things then. We have a drink and we chat. The game play is purely for fun. When we play games on the PC it is much more serious, you need to be very concentrated then, and strictly focused on the game.* (Male participant, FG3, 29 years)

The frequent gamers wrote down several negative emotions as game experiences such as, 'irritation', 'disappointment', 'frustration', and even 'anger'. They reported that frustration and irritation often occur when there is a lack of challenge (i.e., if the game is either too easy or too hard).

*If a game gets too complicated I am often inclined to turn it off, to quit gaming. There has to be some challenge in the game; I don't like it if it is too easy, but if it is too complex I don't like it either. I get irritated if something doesn't work, I sometimes even get angry.* (Male participant, FG3, 23 years)

*I often play RTS games against the same person, if we set a high difficulty setting, it gets more challenging. Of course you feel disappointed if it doesn't work and satisfaction if it does work out. I think disappointment relates to the effort you put into the game.* (Male participant, FG3, 28 years)

Also, negative emotions are stronger if the game play gets more serious.

*Game competitions are dead serious, as serious as a soccer game Holland-Germany. You can really feel aggression, or anger. When you play for fun, it is more informal, having fun is the dominant experience.* (Male participants, FG3, 29 years)

### **Question 3: What do you experience after gaming?**

Frequent gamers were quite unanimous with respect to their experiences after game play. In general, they did not see it as a waste of time and often had the feeling of having done something really useful. Only in very specific situations they reported the experience of disappointment or regret.

*If I play online games I never experience it as a waste of time. When you are cooperating in a team and one member gives up, it is a pity. Then I feel disappointed.* (Male participant, FG3, 23 years)

*Gaming is never a waste of time. Watching TV is much more a waste of time, because it is more passive, you are not involved in what happens.* (Male participant, FG3, 28 years)

*....Only if you have been gaming for quite a long time and you did not achieve anything, I often regret having spent so much time on it. Especially when I have more urgent things to do.* (Male participant, FG3, 29 years)

### 2.2.3. Conclusions

On most themes we found both similarities and differences between frequent and infrequent gamers who participated in our focus groups.

First, both groups reported that they often started gaming when they had little else to do or when they wanted to relax after a busy day. However, the kind of games they played at such occasions seemed to differ. Infrequent gamers mostly chose to play single player games, whereas frequent gamers mentioned quick online FPS games.

Also, in both groups, participants referred to playing games as a way to relax and meet with friends. Here, the type of game did not seem to differ. Moreover, in both groups participants said that on such occasions gaming was a trigger for enjoyment with others and that other factors (e.g., drinking beer, eating chips, having a chat) accompanied the playing.

Notably, only frequent gamers reported coordinated, team play in digital gaming.

With respect to experiences while gaming, both groups reported experiences like fun, enjoyment, concentration, tension, trill and immersion. Nevertheless, the frequent gamers more explicitly linked the different experiences to different types of games. For example, they mentioned that for FPS games feeling immersed in the story of the game is not as important as in MMORPG's. Participants (implicitly) discerned between immersion in the game's story, the game world (i.e., via sensory stimuli), and immersion through game play. Within our focus groups, only frequent gamers referred to experiences like control and prestige.

Frequent gamers also distinguished between experiences while playing games for fun and experiences while playing games in a competition. More specifically, negative emotions were more frequent and stronger for 'serious' gaming. Infrequent gamers reported fewer negative experiences during play.

As for experiences after gaming, frequent gamers were less likely to call it a waste of time. In contrary, for them playing games was a useful activity and often replaced other leisure activities such as sports or watching television. Infrequent gamers more frequently reported regret, especially when gaming made them loose track of time and, as such, neglect more urgent activities.

At present, we have only distinguished between frequent and infrequent gamers. However, there is also considerable variation within each type. In our focus groups this was especially apparent for the infrequent gamers. Some of them only mentioned playing games as a quick activity in between other (often referred to as more useful) activities. As such, this group can be interpreted as 'incidental', or 'casual' gamers. Others reported only playing games with friends, in an informal, social setting, and might be labelled 'social gamers'. (e.g., see: [http://www.gamasutra.com/php-bin/news\\_index.php?story=10681](http://www.gamasutra.com/php-bin/news_index.php?story=10681)). For future research, it would be interesting to further explore differences in game experiences between these different types of gamers.

## 2.3. Expert meeting

After the focus groups described above, an expert meeting was organised, aimed at combining knowledge and insights gathered from theoretical explorations (including the FUGA consortium's deliverable D2.1) and the focus groups, and consolidating these into a tentative model (or rather: concept map) of game experience, and the components it consists of.

Five game researchers participated in this meeting. These included psychologists and experts on measurement development as well as the state of the art in theories of game experience (two females, three males). Two of them are very frequent gamers themselves, the remaining three should be labelled infrequent gamers. This provided a solid base for determining the tentative core dimensions of digital game experience.

The theoretical work performed by all partners in the FUGA project (Deliverable 2.1) provided a firm and broad basis for the development of the scale. In addition, relevant scientific literature was reviewed for any conceptualisations that might complement this work and for existing scales that might serve as a starting point or inspiration for item formulation. In spite of the recent rise of gaming

research, validated questionnaires probing game experience are scarce at this time. Commercial developers of games generally have in-house testers to perform detailed walkthroughs of the game, which provide insight into game mechanics and gameplay. Sweetser and Wyeth (2005) have provided an overview of design heuristics based on the concept of flow and their model has been validated against professional reviews in game review magazines. Although Ermi and Mayra (2005) have developed a self report questionnaire based on their SCI model, their questionnaire is not easily available and thus its validity for our purpose is hard to assess. Their model did provide input for the theoretical framework. Lastly, Ryan, Rigby, and Przybylski (2006) recently developed a new measure of need satisfaction in play, the Player Experience of Need Satisfaction (PENS), elaborated from self-determination theory (SDT, Deci & Ryan, 1985; 2000). A few relevant existing questionnaires probing related psychological concepts (e.g. PANAS, Watson, Clark & Tellegen, 1988; ITC-SOPI, Lessiter et al., 2001; and IMI, Ryan, Mims, & Koestner, 1983, IMI, 2003).

The expert meeting started with an individual and personal reflection of each researcher on what they saw as important game experiences (both in game and post-game experiences). Similar to the procedure of the focus groups, the experts wrote these down on Post-Its. Subsequently, memos were added based on theoretical considerations and the results from the focus group meetings. We then engaged in an interactive and iterative session, organising these on a whiteboard, according to centrality and similarity, thus creating a comprehensive concept map of components of game experience (see Picture 3).



*Picture 3. Constructing the tentative, comprehensive model of game experience.*

The most important components that resulted from this exercise were:

1. Competence: with *accomplishment, euphoria, and pride* as in game experiences, and *pride also as post game experience*.
  2. Flow: with *concentration, being absorbed, flow, loosing track of time, detachment* as in game experiences, and *jetlag* as post game experience
  3. Suspense: with *challenge, pressure, arousal, hoping to win, anxiety, tension* as in game experiences, and *release, feeling empty, exhausted* as post game experiences.
  4. Enjoyment: with *enjoyment, pleasure, relaxation, exuberated* as in game experiences, and *energized* as post game experience.
  5. Sensory immersion: with *sensory immersion and presence* as in game experiences, and *returning to the real world* as post game experience.
  6. Imaginative immersion: with *absorbed in story, empathy, and identification with game character*.
  7. Control: with *control, autonomy, and power*

8. Negative affect: with *shame, anger, irritation, disappointment, ignorance* as in game experiences and *guilt* as post game experience.
9. Connectedness: with *enjoyment with others, being connected with others, empathy* as in game experiences.
10. Negative affective experiences related to playing with others: *jealousy, envy, revenge, guilt*.

This provided the basis for the item generation for the long list of the Game Experience Questionnaire. Although such a list can never be exhaustive, our goal in constructing the list was comprehensiveness rather than selectiveness.

### 3. Development of the Game Experience Questionnaire

In the current chapter we describe the rationale for questionnaire construction and item generation, and the proceed with the methodology, statistical analyses and results of the empirical investigation into the structure of the Game Experience Questionnaire (GEQ). Lastly, all items and subscales are presented and their reliability (internal consistency) is discussed. Sensitivity and validity will be explored in the following chapter.

#### 3.1. General logic of the GEQ development: Core and modular elements

The Game experience questionnaire is developed with a modular structure, consisting of:

1. The core questionnaire. This is the heart of the GEQ, probing multiple components of game experience. Every component should consist of 4-6 items, with high internal consistency. This questionnaire is administered immediately after a gaming session. In addition, a shorter, in-game version of the core GEQ will be developed, which can be administered multiple times during a gaming session (e.g., during short breaks). This in-game version will be modelled after the core questionnaire, but consists of only two items per component.
2. The post-game questionnaire, probing the gamers' experience after having played the game and any after effects (e.g., returning to reality, fatigue, pride, guilt).

These first two questionnaires should be applicable for any type of digital game played, in any type of setting. The remaining modules are developed for specific types of games or settings. One such module is:

3. The social presence module, probing the gamers' experience of and involvement with their co-player.

This module only applies to situations where digital gamers played with or against one or more social entities, be they co-located players, online players, or virtual players (i.e., in-game characters). In the near future, additional modules will be developed. We envisage:

4. Embodiment module: physical engagement, kinaesthetics, embodied play).
5. Pervasive game module (mixed reality, city games).

According to the present plans as stated in D3.1, these modules will not be applied within the FUGA project in the near future. We have therefore decided to first focus on the development of the core module, the post-game module, and the social presence module.

#### 3.2. Item generation

The results of the focus groups and expert meeting were used as a starting point and reference guide for item generation. A long list of items was created with the following requirements:

1. Items should probe experience 'sec', i.e., is possible, not refer to specific game characteristics, events, or external circumstances, nor should it require respondents to attribute experiences to specific circumstances or events.
2. Wording should be as natural and intelligible as possible for all types of research participants. In formulating the items we tried to stay close to the original wording used by participants of the focus groups.
3. Qualifiers: the use of qualifiers in the items should be minimized, if not prevented.
4. Items should be formulated such that one uniform type of answering scale can be applied, in order to minimise participant burden.

A 5-point, unipolar intensity-based answering scale was developed, with points anchored at *not at all* (0), *slightly* (1), *moderately* (2), *fairly* (3), and *extremely* (4), thus qualifying the intensity of the experience described in the item.

In total, 92 items were formulated in the long list for the core GEQ. The list was constructed such that all ten components of the tentative model, listed in section 4.3. were represented by at least five or more items. Any other possibly relevant items probing additional aspects of game experience were also added, mainly based on focus group data. A few relevant existing questionnaires probing related psychological concepts (e.g. PANAS, ITC-SOPI, IMI) were also used. Twenty one items were formulated for the post-game module.

For the social presence questionnaire, Biocca et al.'s (2001) networked minds measure of social presence provided an excellent starting point for item formulation. It is based on a recently developed theory of social presence and constructed for measuring experiences in social presence technologies such as mediated collaborative work environments, mobile and wireless telecommunication, and teleconferencing interfaces (Biocca et al. 2003). Some items were used directly, others were adapted and some new items were formulated based on the focus group data. The long-list consisted of 25 items. The answering scale used was identical to that used for the GEQ and PGQ.

### 3.3. Research method

#### Design

For exploring the factor structure of our original set of items a sizeable and representative sample of gamers was needed. As the questionnaire measure is intended to be applicable across a range of gaming genres and platforms, it was important to allow participants to play a self-selected game on their own preferred gaming platform (PC, console, mobile), and in their own preferred physical setting (e.g., living room, bedroom, Internet Café, etc.). Based on these considerations we chose to use an online survey, inviting people to play their game of choice and to fill out our questionnaire immediately afterwards.

All items from the game experience questionnaire (see section 5.2), consisting of the core questionnaire (92 items), the social presence questionnaire (25 items), and the post-game questionnaire (21 items), were measured by means of a five point intensity scale with points anchored at *not at all* (0), *slightly* (1), *moderately* (2), *fairly* (3), *extremely* (4).

The full online survey was set up in four parts: one part with instructions, an introduction, the actual game experience questionnaire, and a final part with general game related and socio demographic questions. We will elaborate on those parts in the procedure section.

#### Participant recruitment and selection

Participants were recruited via the internet, using two different channels: Virtual Lab (vlab) and internet game forums. Both websites allowed us to track the time participants spent in filling out the questionnaire and allowed us to immediately exclude users who had already filled in the questionnaire once. On average, it took ten to 15 minutes to complete the full survey. If participants were exceptionally fast at filling out the questionnaire (i.e., eight minutes or less), we discarded them from the dataset. If they were faster than average (i.e., less than ten minutes), we checked whether they were consistent in their answers by looking at the correlation of three fun-related variables (*I enjoyed it, I thought it was fun, I liked it*). If these were not or negatively correlated, we excluded those participants from our final sample.

We launched our survey at vlab, an internet based lab connected to the Human Technology Interaction Department of Eindhoven University of Technology. An invitation to participate in a study on game experience was sent out to 495 people registered at the vlab database. A total of 262 people responded to this invitation, yielding an initial response rate of 53%. However, we excluded 50 respondents who did not fill in all game experience items or who filled in the questionnaire either too quickly or inconsistently from our main sample. Respondents who participated through vlab were paid 3 €. We further recruited participants by posting invitations at several Dutch and Belgian internet game forums (e.g., InsideGamer.nl, Minatica.be). The invitation included a link to our survey created with the online survey tool SurveyMonkey ([www.surveymonkey.com](http://www.surveymonkey.com)). As an incentive, we raffled a PlayStation 3 among participants who took part through this online link. Two hundred and ten people reacted to this invitation. Forty two of them filled in the questionnaire too quickly, incompletely and/or inconsistently and were therefore excluded from our final sample.

## Participants

Our final sample consisted of 380 participants who played a game of their own choice. This group consisted of 254 men and 120 women (sex value missing = 6), with an average age of 20.8 years (range 10 to 61 years,  $SD = 5.26$  years). With respect to educational level, 5% had a low education, 13% a mid level education, and 81% was highly educated. Gaming frequency varied from daily (29%), at least weekly (38%), at least monthly (13%), at least a few times per year (12%), and hardly ever (8%). Participants that never played a digital game were excluded from the current sample.

## Game characteristics

The type of games participants played, were myriad. Participants filled in the full name of the game and, with the help of a game expert, we recoded those games into 12 game genres. Participants played First Person Shooter games (22%), Role Playing games (14%), Sport games (13%), Puzzle/board/card games (11%), Action adventure games (10%), Strategy games (9%), and other genres (e.g., simulation games, fight games, children's games, music games) (11%).

Most games were played on the PC (62%) or on a console (32%). A minority (6%) used another platform (e.g., handheld console or mobile phone). Further, the majority of participants played the game inside the house: in their bedroom (43%), in the living room (29%), in their study room (18%). A minority (10%) played outside the house (e.g., public transport, car, pub).

With respect to social setting; 73% played alone, 20% played online with others, and 6% played with co-located others. From the participants who played alone, 38% played against virtual others.

Game frequency of the game they reported on ranged from less than once a month (20%), monthly (18%), weekly (41%), to daily (22%). Experience with the game varied from the first time they played that game (8%), between one week and one month of experience with that game (22%), between one month and one year of experience with that game (29%), and more than one year of experience with that game (40%). The duration of the game session varied from less than half an hour (21%), between half an hour and one hour (33%), between one hour and three hours (40%), and more than three hours (6%).

## Procedure

First, as outlined above, participants were invited to take part in a study on game experience. The invitation described the purpose and the procedure of the study. More concretely, we told participants that we were interested in how people experience digital gaming and that everybody, also non frequent gamers, could participate. In the invitation we further included the instruction that before opening the link to the questionnaire they had to play a game. Participants could freely chose the game they played. However, we suggested that they would best play a game in the way they normally do (with regard to the type of game, gaming platform, physical game setting). After playing the game, participants could click on a link that guided them to the online survey. This survey started with an introduction in which we described the course of the questionnaire. Next, in the introductory part, participants had to report some details about the game they just played (e.g. name of the game, gaming device, single or multiplayer, location, experience with the game, duration of the game). Subsequently, participants were asked to report on their game experience by completing the core game experience questionnaire, the social presence questionnaire, and the post-game questionnaire. The social presence questionnaire was only shown to people who had indicated that they had played the game with other people or mediated (e.g., online) others. We ended the survey with some general questions concerning people's gaming behaviour (game frequency, average duration of a game session, how much they engage in several types of single and multiplayer game settings, type of games they play, gaming motivations, proportion of leisure time spent on gaming) and some basic socio demographic questions (sex, age, education, media behaviour).

### 3.4. Exploratory factor analysis and scale reliability

#### 3.4.1. Rationale for statistical procedure

Exploratory Factor Analysis (EFA) was performed to investigate the component structure of the Game Experience Questionnaire. Ideally, a theoretical model of game experience would prescribe the structure of such a questionnaire. However, although theoretical and empirical explorations

described earlier do suggest the relevance and existence of certain components, this basis is not firm enough to warrant the use of confirmatory factor analysis. In performing EFA, important decisions have to be taken on a number of methodological issues, such as, (1) what items to include, (3) what procedure to use to fit the model to the data, (3) what number of factors to choose, and (4) what rotation method to use on the initial factor solution (Fabrigar, Wegener, MacCallum, Strahan, 1999). Unfortunately, no strict rules for making these decisions exist, and most of them are a matter of continuous debate among scholars. In our analyses, we have tried to adhere closely to suggestions and advice as formulated by Fabrigar and colleagues (1999), Tabachnick and Fidell (2001), and Costello and Osborne (2005).

Ad 1: As we reported earlier, item selection was based on thorough theoretical and empirical work. Items were subsequently formulated to match the wording used by gamers – as recorded during focus groups.

Ad 2: Principle axis factoring was selected, as this technique is best suited to explore the structure of latent variables behind a collection of items measuring a multi-dimensional concept, such as game experience, in contrast to for instance PCA which is aimed to explain a maximum proportion of variance in the data.

Ad 3: Several criteria exist for determining the optimal number of factors in factor analysis. Some have proposed the Kaiser criterion, where all factors with an eigenvalue  $> 1$  should be selected. This criterion has met with considerable criticism. An alternative is the scree-plot criterion: when eigenvalues are plotted in order of descending values in a graph, the optimal number of factors is indicated by a sharp bend or break point in the data where the curve flattens out. In addition, several goodness-of-fit indexes have been proposed for determining the optimal number of factors. Lastly, Fabrigar et al. (1999) argue: 'the decision of how many factors to include in a model is a substantive issue as well as a statistical one' (p. 281). Theoretical and conceptual considerations of multiple test runs (i.e., with different factor solutions) are therefore a valuable and valid step in the procedure (see also Costello & Osborne, 2005). The resulting model should be interpretable and theoretically sensible.

Ad 4: The last important decision to make in the EFA procedure is what type of rotation to use. The most basic choice is between orthogonal and oblique rotations. Orthogonal rotations – e.g. varimax – constrain factors to be uncorrelated. In contrast, oblique rotations – such as, oblimin – permit correlations among factors. Varimax is probably the most well-known and widely used in psychological research. However, components of game experience are very likely to be correlated with one another. If the latent variables are, in fact, correlated, then an oblique rotation will produce a better estimate. Following comments by Fabrigar et al., Tabachnik and Fidell, and Costello and Osborne, we decided to use oblimin as a first choice, but also explored the varimax rotated factor solutions, which always resulted in very similar factor solutions.

Subsequent analyses involved scale reliability analysis, with the general aim of selecting five or six strongly loading items for every factor we included in the GEQ and resulting Cronbach's alpha scores acceptable to good, i.e. 0.7 or higher. Selection of items was based on their individual factor loadings and their correlations with the scale. Items with cross-loadings above 0.30 were not selected for inclusion in the scales.

### **3.4.2. GEQ – core**

To analyse the structure of the core part of the GEQ, exploratory factor analysis (EFA) was performed on the collected data.

Of the items originally used in the long list, some were discarded based on their lack of variance (e.g., I felt bad, It gave me a bad mood, I found it tiresome, I felt angry, I felt ashamed). The resulting set of 82 items was used in exploratory factor analysis. Considering the number of research participants, this resulted in a item-participant ratio of about 1: 5.

The initial factor analysis revealed 14 factors with an eigenvalue over 1. On the other hand, the scree plot showed that after the first five, or perhaps eight components, eigenvalues started to decrease asymptotically. We then explored the 5, 6, 7 and 8-factor solutions in more detail. The 5-factor solution was very clear and clean, but so was the 7-factor solution, which rendered two additional factors which were hypothesised and easily interpretable. As a comparison, we ran a varimax rotation on the 7-factor solution, which was almost identical and produced the same components. As subsequent scale analyses demonstrated that these latter factors produced very reliable scales, we decided upon the 7-factor solution. Several scholars argue that specifying too few factors in a model

(i.e., underfactoring) is a more severe error than specifying too many (i.e., overfactoring) (Fabrigar et al., 1999).

The 7-factor solution explained 52% of variance in the total number of items. All factors were very clear, i.e., almost self-explanatory and most items loading on only one factor (criterion .30). The seven components are:

1. Sensory and Imaginative Immersion
2. Tension
3. Competence
4. Flow
5. Negative Affect
6. Positive affect
7. Challenge.

The factor solution and scale construction presented here provide a clear and clean structure for the Game Experience Questionnaire. The structure closely resembles the structure proposed in the tentative model, with a few exceptions. First, sensory and imaginative immersion are combined into one single scale. In addition, no subscale emerged that probes experienced control. Some of the items envisaged for this scale appear in the competence scale. Competence and control are psychologically related concepts. Also, an unforeseen component emerged probing ‘tension’, separate from other negative affect. The last two components listed in Section 4.3 both appear in the social presence module, with an additional third component, called ‘behavioural involvement’.

Scores for subscales are computed as the mean score of the items in the scales. The full questionnaire is listed in Appendix 1, in its final order.

### **Sensory and Imaginative Immersion**

The first factor that emerged is called *Sensory and Imaginative Immersion*. Originally consisting of 23 items, subsequent scale reliability analyses resulted in the selection of the following items, with a Cronbach's alpha = 0.891, and factor loadings ranging from .587 - .847. Note: as an exception, 6 items were selected, three referring more to the sensory aspect, the other three to the more imaginative part of immersion. For the remaining factors we propose five items per scale (original Dutch items are given in parentheses).

I was aesthetically pleasing (*Ik vond het aansprekend van vormgeving*)

I was interested in the game's story (*Ik was geboeid door het verhaal van het spel*)

I felt imaginative (*Ik voelde me fantasierijk*)

I felt that I could explore things (*Ik had het gevoel dat ik de gamewereld kon exploreren*)

I found it impressive (*Ik vond het indrukwekkend*)

It felt like a rich experience (*Ik vond het een rijke ervaring*)

### **Tension**

The second factor that emerged was termed *Tension*. From the twelve items loading high on this factor, five were included in the final Dutch scale, and one additional item was selected as a spare item for translation purposes. The resulting scale had an internal consistency of 0.811 (Cronbach's alpha; with extra item 0.821), and factor loadings ranging between .606 and .704.

I felt tense (*Ik voelde me gespannen*)

I felt restless (*Ik voelde me onrustig*)

I felt annoyed (*Ik was geïrriteerd*)

I felt frustrated (*Ik was gefrustreerd*)

I felt irritable (*Ik was prikkelbaar*)

I felt pressured (*spare item*)

### Competence

The third factor was termed Competence. Sixteen items had factor loadings above .30 on this factor. Again, five were selected for the Dutch scale, plus one extra for translation purposes. The resulting Cronbach's alpha = 0.826 (with extra item 0.843), and factor loadings ranging from .474 to .755.

I felt strong (*Ik voelde me zeker*)

I was good at it (*Ik was er goed in*)

I felt successful (*Ik voelde me succesvol*)

I felt skilful (*Ik voelde me vaardig*)

I was fast at reaching the game's targets (*Ik was snel in het bereiken van de doelen in de game*)

I felt competent (*extra*)

### Flow

The fourth factor that emerged was *Flow*. From the thirteen items loading high on this factor, five plus one were selected. The Flow-scale has an internal consistency score of 0.866 (alpha with extra item 0.883), and factor loadings between .614 and .821.

I forgot everything around me (*ik vergat alles om me heen*)

I felt completely absorbed (*Ik was helemaal geabsorbeerd*)

I lost track of time (*Ik was mijn gevoel voor tijd kwijt*)

I lost connection with the outside world (*ik was weg uit de buitenwereld*)

I was deeply concentrated in the game (*Ik was ten volle geconcentreerd op de game*)

I was fully occupied with the game (*extra*)

### Negative Affect

The fifth factor – *Negative Affect* – was actually the most problematic one. Six items loaded high on this factor, but three of them had higher cross-loadings on other factors. In fact, all factor loadings were rather low on this factor. However, we feel that this may be an artefact, created by our instructions to participants, to first play one of their own favourite games and then fill in the questionnaire. This implies that most participants will have enjoyed their game and that they knew how to play it. In preparing for the analysis, several of the items discarded due to low variance and skewed distributions probably belonged to this group of items. As in future the questionnaire will also be used for probing people's experience of games they are not familiar with and possibly dislike, having a module measuring negative affect does seem crucial. We therefore went on to explore scale reliability using two items that were originally discarded (see starred items) and found the following set with good interpretability, theoretically sensible, and with a very acceptable Cronbach's alpha value of 0.712 (0.728 with the extra item). The reliability of this subscale will be explored further with new datasets.

I thought about other things (*Ik was met andere zaken bezig*)

I felt bored (*Ik voelde me verveeld*)

I was distracted (*Ik was afgeleid*)

I was bored by the story (*Ik vond het verhaal van het spel saai*)

I found it tiresome (*Ik vond het saai*)\*

It gave me a bad mood (*extra*)\*

### Positive Affect

The sixth scale is termed *Positive Affect* and basically probes the fun and enjoyment of gaming. Eight items loaded on this factor, of which five plus one were selected. Cronbach's alpha for the five-item scale was 0.797 (0.838 with extra item). Factor loadings varied between .430 and .728.

I felt happy (*Ik voelde me vrolijk*)

I felt content (Ik voelde me tevreden)  
I felt good (Ik voelde me lekker)  
I enjoyed it (Ik genoot ervan)  
I could laugh about it (Ik kon er om lachen)  
I thought it was fun (extra)

### Challenge

The seventh and last factor is termed *Challenge*. All six items that loaded on this factor were selected for the scale. The sixth item cross-loaded on factors one and two and was therefore not selected in the final Dutch scale, but is suggested as spare item for translation purposes. Cronbach's alpha for the five-item scale is 0.745 (0.718 for the six-item scale). Factor loadings varied between .605 and .348.

I felt challenged (Ik voelde me uitgedaagd)  
I thought it was hard (Ik vond het moeilijk)  
I had to put a lot of effort into it (Ik moest er veel moeite in steken)  
I felt that I was learning (Ik had het gevoel dat ik aan het leren was)  
I felt stimulated (Ik voelde me gestimuleerd)  
I felt time pressure (extra)

### 3.4.3. In-game questionnaire (iGEQ)

For validation purposes of continuous (real-time) indicators of game experience, several partners have indicated the need for a concise 'in-game' version of the GEQ. This will be used for probing Game Experience at regular intervals during the game, and for assessing multiple short-duration phases or game events in play-back mode. For the in-game questionnaire, an identical factor structure is chosen as for the core GEQ. For every component, two items were chosen which:

1. had acceptable (or better) alpha scores, and
2. semantically represented the component's item best.

Cronbach's alpha for the two-items scales varied between 0.693 and 0.80. The in-game GEQ (short form) is reported in Appendix 2.

### 3.4.4. Post-game questionnaire (PGQ)

The post-game module is aimed towards measuring how people feel after they have stopped playing. Since we requested participants in our study to play a game of their own choice before filling in the questionnaire, we realise that this may have created a strong bias away from negatively termed items such as, uselessness, guilt, shame, or regret. After all, they *had* to play in order to fulfil their 'job'. In line with this consideration, we found strongly skewed data on these and related items, resulting in low variances. This artefact would result in discarding quite a high – disproportionate – number of items from the analysis. We therefore decided not to discard items prior to analysis and regard the findings on this scale as preliminary. Evidently, additional research is needed to explore and validate the structure of this module. As a side note: this module is not crucial to experimentation work in the project. We will however, present the preliminary results in this report.

Exploratory Factor Analysis (EFA) was performed on the full set of 21 items probing post-game experience. Initial PFA resulted in four factors with an eigenvalue higher than one. Although the screeplot seemed to indicate two as an optimal number of factors, we selected all four, since they represented interpretable and clear factors. Subsequent oblique rotation resulted in the following structure.

The structure of the post-game module presented here should be regarded as preliminary. Future studies will further explore and test components and subscale reliabilities. The module is reported in Appendix 3.

### Negative experiences

The first factor that emerged is termed *Negative experiences*. Of the eight factors loading high on this component, six items were selected for the preliminary scale, with factor loadings ranging between .591 and .762, resulting in a scale reliability of 0.832.

I found it a waste of time (*Ik vond het zonde van de tijd*)

I felt that I could have done more useful things (*Ik vond dat ik meer nuttigs had kunnen doen*)

I felt regret (*Ik had spijt*)

I felt guilty (*Ik voelde me schuldig*)

I felt ashamed (*Ik schaamde me*)

I felt bad (*Ik voelde me rot*)

### Positive Experiences

The second factor was termed *Positive experiences*. Six items of the original eight were selected to form a scale with Cronbach's alpha of 0.900, with factor loadings ranging between .678 and .854.

I felt like a victory (*Ik zag het als een overwinning*)

I felt proud (*Ik voelde me trots*)

I felt powerful (*Ik voelde me machtig*)

I felt satisfied (*Ik voelde me voldaan*)

I felt revived (*Ik voelde me opgepept*)

I felt energised (*Ik voelde me opgeladen*)

### Tiredness

The third factor only consisted of two items with factor loadings above .40 (.926 and .676 respectively), and a Cronbach's alpha score of 0.764. The two items are:

I felt exhausted (*Ik voelde me uitgeput*)

I felt weary (*Ik voelde me vermoeid*)

### Returning to Reality

The fourth and last factor consisted of three items with factor loadings between .404 and .702. These three items had an internal consistency (Cronbach's alpha) of 0.619. This factor, called *Returning to reality*, consisted of the following items:

I found it hard to get back to reality (*Ik had moeite om terug te keren naar de realiteit*)

I felt disoriented (*Ik voelde me gedesorienteerd*)

I had a sense that I had returned from a journey (*Ik had het gevoel dat ik van een reis terugkeerde*)

### 3.4.5. GEQ – social presence module

Although a number of items used in the social presence module were taken from or inspired by the Networked Minds measure of Social Presence (Biocca et al., 2001), some items were new to this scale. In addition, the application area of gaming differs significantly from the application area. We therefore decided to also perform exploratory factor analysis (EFA) on this set of items.

Of the 25 items originally used in the long list, again five had to be discarded based on their lack of variance. The resulting set of 20 items was used in exploratory factor analysis. The sample size for this analysis was N=212, which is smaller than in the previous analyses, as it only consisted of participants who indicated to have played with or against virtual, mediated, or co-located others. This resulted in a item-participant ratio of about 1: 10.

The initial factor analysis revealed five factors with an eigenvalue higher than one. However, the scree plot showed a clear bend after the third factor. The three-factor solution was very clear, and

explained 56% of variance. The solution resulted in two psychological involvement components (Empathy and Negative feelings), and a behavioural involvement component.

The resulting structure is quite different from the one in Biocca et al's networked minds measure of social presence. This is not surprising as the application domains differ considerably. For instance, the very act of playing a game together implies that players' actions are dependent on each other, even in conditions of minimal 'media richness'. Behavioural interdependence is one of the hard to reach targets in application domains such as teleconference systems.

Together these items make up the social presence module, which is only to be used in gaming situations where players played with or against others, be they virtual (e.g., in-game characters), mediated (e.g., online), or co-located. The module is reported in Appendix 4.

### **Psychological Involvement - Empathy**

The first factor that emerged is called *Psychological Involvement - Empathy*. Six of the seven items loading on this factor were selected for the scale, with a Cronbach's alpha = 0.886, and factor loadings ranging from .683 - .826.

I empathized with the others (*Ik leefde mee met de anderen*)

When I was happy, the others were happy (*Wanneer ik blij was, waren de anderen blij*)

When the others were happy, I was happy (*Wanneer de anderen blij waren, was ik blij*)

I felt connected to the others (*Ik voelde me verbonden met de anderen*)

I admired the others (*Ik bewonderde de anderen*)

I found it enjoyable to be with the others (*Ik vond het gezellig met de anderen*)

### **Psychological Involvement – Negative feelings**

The second component of *Psychological Involvement* is called *Negative feelings*. Again, six of the original seven items were selected for the scale, which has an internal consistency (alpha) of 0.860, and factor loadings between .511 and .855.

I was influenced by the other's moods (*Ik werd beïnvloed door de stemming van de ander*)

I influenced the mood of the other (*Ik beïnvloedde de stemming van de ander*)

I felt revengeful (*Ik was wraakzuchtig*)

I felt schadenfreude (malicious delight) (*Ik had leedvermaak*)

I felt jealous of the other (*Ik was jaloers op de ander*)

### **Behavioural Involvement**

The third scale is termed *Behavioural Involvement*. All five items that loaded on this factor were selected for the scale, with a Cronbach's alpha of 0.711, and factor loadings between .338 and .774. The items are:

My actions depended on the other's actions (Mijn handelingen hingen af van de handelingen van de ander)

The other's actions were dependent on my actions (De handelingen van de ander hingen af van mijn handelingen)

What the others did affected what I did (*Wat de ander deed beïnvloedde wat ik deed*)

What I did affected what the other did (*Wat ik deed, beïnvloedde wat de ander deed*)

The other paid close attention to me (*De ander lette op mij*)

I paid close attention to the other (*Ik lette op de ander*)

### 3.5. Conclusion

In this chapter the rationale for questionnaire construction and item generation were discussed. Subsequently, the methodology and statistical analyses and results of the empirical investigation into the structure of the Game Experience Questionnaire (GEQ) were described.

The Game experience questionnaire is developed with a modular structure, consisting of:

1. The core questionnaire (GEQ). This is the heart of the Game Experience Questionnaire, probing multiple components of players' experience while gaming.
2. The post-game questionnaire (PGQ), probing gamers' experience after the gaming session and any after effects.
3. The social presence module (SPGQ), probing gamers' experience of and involvement with their co-player(s).

These three lists are to be administered after the gaming session has ended. Additionally, a short in-game version of the GEQ was developed, the iGEQ, for probing in-game experience multiple times during a gaming session.

For the core GEQ, the factor solution and scale analyses provided a clear and clean structure of seven components (1) Sensory and Imaginative Immersion, (2) Tension, (3) Competence, (4) Flow, (5) Negative Affect, (6) Positive affect, and (7) Challenge. These subscales, each reliably measuring unique components of game experience, have 5 or 6 items, and a good average Cronbach's alpha score of .81.

Scale reliabilities of a short, in-game version of the GEQ for validation purposes of continuous indicators of game experience were also explored. The iGEQ's factor structure is identical to that of the core GEQ, but consists of only 2 items per scale, selected to best represent the psychological concept. Results showed that these scales were also reliable, with internal consistencies varying between satisfactory and good.

The post-game module (PGQ), aimed towards measuring how people feel after they have stopped playing, consists of four subscales. As the analysis did not unanimously and clearly indicate one optimal structure for this module, the structure presented here should be regarded as preliminary. Future studies will further explore and test components and subscale reliabilities. The next chapter will test their validity and sensitivity.

Dimensionality analysis of the social presence gaming questionnaire (SPGQ) resulted in three subscales: (1) Psychological involvement – Empathy, (2) Psychological Involvement – Negative feelings, and (3) Behavioural involvement. As gaming presents a different application domain from the domains targeted Biocca et al. (2001), the resulting structure differs from the one presented there. Reliabilities of the scales are good, average alpha .82. The scales are only to be used in gaming situations where players played with or against others, be they virtual (e.g., in-game characters), mediated (e.g., online), or co-located.

Scores for all subscales are computed as the mean score of the items in the scales. The full questionnaires are listed in Appendix 1 - 4. Scoring guidelines are reported in Appendix 5.

The results of the analyses presented in the present chapter are promising, as they have mostly rendered reliable and easily interpretable scales, matching theoretical conceptualisation and qualitative empirical insights. Sensitivity and validity of the scales constructed and discussed here will be explored in the following chapter.

## 4. Explorations of the GEQ: Inter-correlations and basic sensitivity

Although the structure of the GEQ presented in Chapter 3 is clear and easily interpretable, internal consistencies of all subscales are acceptable or better, and both face validity and content validity are high, additional explorations are needed to check whether the developed measure is sensitive enough to pick up differences between games, game characteristics, gamers, and game circumstances. Part of this work will be performed as an integral part of the experimental work planned in the FUGA project (see D3.1). However, the present chapter describes preliminary explorations, based on the data gathered in the survey study reported earlier. In each section below, basic sensitivity and discriminant validity of subscales in each of the GEQ modules are reported and discussed, based on background data that was gathered from respondents.

### 4.1. Core Game Experience Questionnaire (GEQ)

The core GEQ consists of seven subscales, each measuring unique, though sometimes related components of game experience. Basic descriptives for the subscales are reported in Table 1. Scores on most scales were moderate and ranged the full scale. The highest scores appeared on Positive Affect and Immersion. Scores for Negative affect and Tension were rather low.

**Table 1.** Basic descriptives for GEQ subscales.

|                 | Mean | SD   | Min | max  |
|-----------------|------|------|-----|------|
| Positive affect | 2.55 | .73  | .00 | 4.00 |
| Competence      | 1.72 | 1.12 | .00 | 4.00 |
| Immersion       | 2.28 | .80  | .00 | 4.00 |
| Flow            | 1.63 | 1.02 | .00 | 4.00 |
| Challenge       | 1.70 | .79  | .00 | 4.00 |
| Negative affect | .57  | .60  | .00 | 3.40 |
| Tension         | .88  | .77  | .00 | 3.80 |

Note: scales range from 0 to 4

Correlations between the subscales are reported in Table 2. As could be expected, there are a number of significant correlations between subscales of the GEQ. The highest correlations (close to .5) exist between Positive Affect, Competence, and Immersion. Flow is correlated moderately with Competence, Challenge, and Immersion. Negative Affect is negatively correlated with all subscales but the Tension scale. This Tension scale has modest (.2 -.3) positive correlations with Challenge, Flow, and Negative Affect, and a small negative correlation with Positive Affect.

Sensitivity of Core GEQ subscales will be explored by investigating gender differences, differences between frequent and infrequent gamers, type of game played, and social setting.

**Table 2.** Bivariate correlation between GEQ subscales.

|                 | Positive affect | Immersion | Competence | Flow   | Challenge | Negative affect | Tension |
|-----------------|-----------------|-----------|------------|--------|-----------|-----------------|---------|
| Positive affect |                 | .467*     | .514*      | .278*  | .340*     | -.462*          | -.146*  |
| Immersion       | .467*           |           | .519*      | .488*  | .472*     | -.432*          | .054    |
| Competence      | .514*           | .519*     |            | .437*  | .395*     | -.313*          | .053    |
| Flow            | .278*           | .488*     | .437*      |        | .454*     | -.317*          | .217*   |
| Challenge       | .340*           | .472*     | .395*      | .454*  |           | -.285*          | .317*   |
| Negative affect | -.462*          | -.432*    | -.313*     | -.317* | -.285*    |                 | .162*   |
| Tension         | -.146*          | .054      | .053       | .217*  | .317*     | .162*           |         |

Note: N = 380; \* = p < .01

### Gender differences

About one-third of participants in the study's sample are female. Analyses of variance showed that female players report significantly lower scores ( $p < .001$ ) on every single subscale of the GEQ, except for Negative Affect, where females score higher, and Tension, where there is no significant difference. Mean scores are reported in Table 3.

**Table 3.** Gender differences on GEQ subscales

|        | Positive affect | Immersion   | Competence | Flow        | Challenge  | Negative affect | Tension    |
|--------|-----------------|-------------|------------|-------------|------------|-----------------|------------|
| Female | 2.32 (.67)      | 1.09 (.93)  | 1.96 (.81) | 1.30 (1.03) | 1.39 (.77) | 0.77 (.64)      | 0.79 (.78) |
| Male   | 2.67 (.71)      | 2.02 (1.08) | 2.45 (.74) | 1.80 (.97)  | 1.84 (.76) | 0.48 (.56)      | 0.92 (.75) |
| Total  | 2.56 (.72)      | 1.72 (1.12) | 2.29 (.80) | 1.64 (1.02) | 1.70 (.79) | 0.57 (.60)      | 0.88 (.76) |

Note: N females = 120, N males = 254.

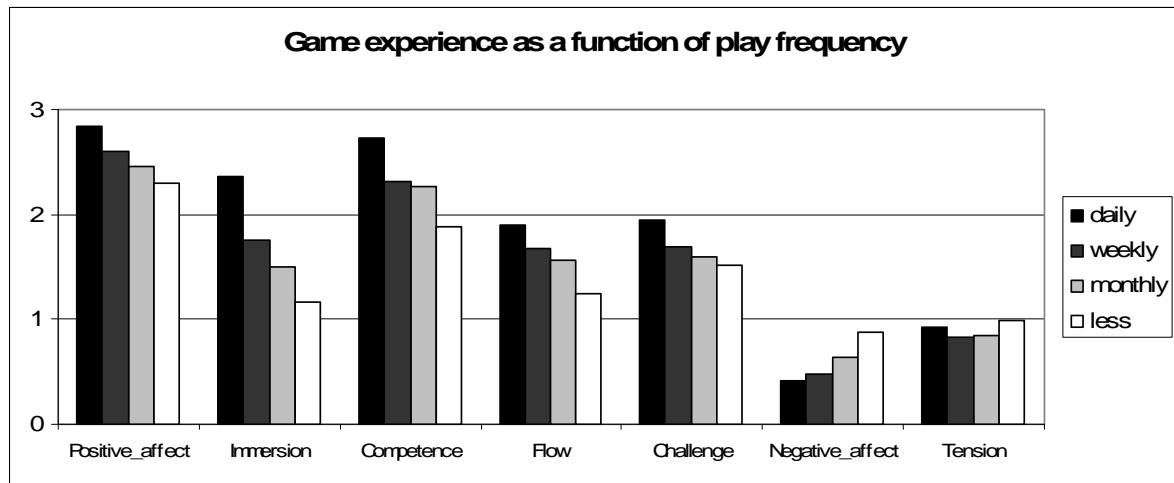
### Frequent vs. infrequent gamers

We explored game experience as a function of how often participants played the game they reported on. Positive Affect, Immersion, Competence, Flow, and Challenge all increased with play frequency, as would be expected for all scales, especially the first four. Scores are visualised in Figure 1. Analyses of variance showed that these differences were significant (for Challenge,  $p = .005$  all remaining  $p < .001$ ). Subsequent contrast analyses even indicated that for Positive Affect, Immersion, Competence and Challenge, all differences between all groups were significant at the .05 level. Negative Affect showed an inverse relation with play frequency. Reported tension was not related to play frequency.

### Type of game

We subsequently explored whether game experience dimensions varied as a function of the type of game participants played. For most game experience components, analyses of variance yielded significant differences (all  $p < .05$ ) for Type of game played. Scores for each game type on the game experience dimensions, are visualised in Figure 2. Contrast analyses were performed to test specific differences between game types.

Positive affect showed significant differences between game types,  $F(5, 302) = 4.89$ ,  $p < .001$ , Puzzle games scored significantly lower than all other types of games. The reverse was true for Negative Affect,  $F(5, 302) = 10.36$ ,  $p < .001$ . Similar patterns emerged for Competence  $F(5, 302) = 2.59$ ,  $p = .03$ , for which Puzzle games scored lower than FPSs and strategy games, and Flow,  $F(5, 302) = 2.86$ ,  $p < .02$ .



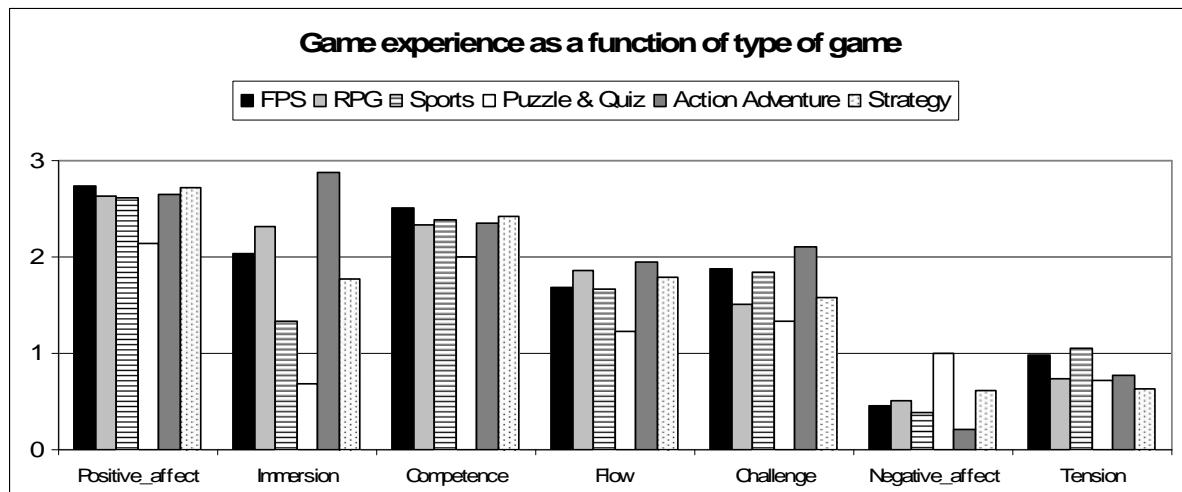
**Figure 1.** Game experience on GEQ subscales as a function of play frequency (this particular game)

Game type showed significant effects on Imaginative and Sensory Immersion,  $F(5, 302) = 32.20$ ,  $p < .001$ . Action adventure games scored significantly higher than all other types of games. Role playing games (RPG) and first person shooter games (FPS) did not differ from each other but scored higher than sports games, strategy games and puzzle games. Subsequently, both sports and strategy games scored higher than puzzle games.

Action adventure, sports and FPS games scored highest on the Challenge dimension,  $F(5, 302) = 7.00$ ,  $p < .001$ . Action adventure games differed significantly from all remaining types of games (RPG, puzzle and strategy); FPS and Sports games differed only from puzzle games.

Post hoc tests revealed no significant effects between the different game types on the Tension subscale,  $F(5, 302) = 2.4$ ,  $p = .04$ .

Note: the number of participants for every type of game was rather low ( $N$  varied from 34 to 87) which could explain why some post hoc tests failed to reach significance. Nevertheless, there are some substantial differences between the different types of games that certainly deserve further exploration.



**Figure 2.** Game experience on GEQ subscales as a function of type of game

### Social setting

To further explore the sensitivity and discriminant validity of the subscales, GEQ scores were compared for various social settings in which participants played. Based on participants' responses on two scales from the background questionnaire we constructed a categorisation of increasing social presence and relatedness. The categories describe settings in which the 'distance' (physically & psychologically) between players and co-players gets smaller. The 5 categories are:

- (1) playing alone,
- (2) playing with virtual others (i.e., in-game characters),
- (3) playing online with unknown others,
- (4) playing online with friends/family,
- (5) with the co-player physically present.

The analyses are performed on a slightly smaller dataset than the original one, as the categorisation could not be unambiguously made for 27 participants. As the number of conflicting answers on the two scales used was rather high (27 out of 380), the results should be regarded with some caution. Some ambiguity may have existed in these scales, possibly weakening the reliability of this variable.

GEQ scores for the five different groups are visualised in Figure 3. All subscales except Flow and Tension showed significant differences between social settings. Moreover, score patterns for the subscales differed, indicating the distinctiveness of the components measured (i.e., discriminant validity).

Positive affect differed significantly between groups,  $F(4, 348)= 7.04$ ,  $p<.001$ , with significant contrasts between the 1<sup>st</sup> and 2<sup>nd</sup>, the 3<sup>rd</sup> and 4<sup>th</sup>, and the 4<sup>th</sup> and 5<sup>th</sup> category. Participants reported higher enjoyment with each subsequent social setting of increased social presence and relatedness.

Sensory and Imaginative Immersion showed a significant effect of social setting,  $F(4, 348)= 2.36$ ,  $p=.05$ . Interestingly, contrast analyses indicated that this should be attributed solely to a significant difference between the 1<sup>st</sup> and 2<sup>nd</sup> category. In both groups, participants played alone, either without (group 1) or with (group 2) in-game characters. No significant contrasts emerged between remaining categories. Perhaps the effect is caused by the different types of games played by gamers in these two groups. For instance, participants in the first group more frequently played puzzles – which are generally experienced as less immersive – whereas participants in the second group more frequently played FPS and sports games, which are generally experienced as highly immersive (see Figure 2).

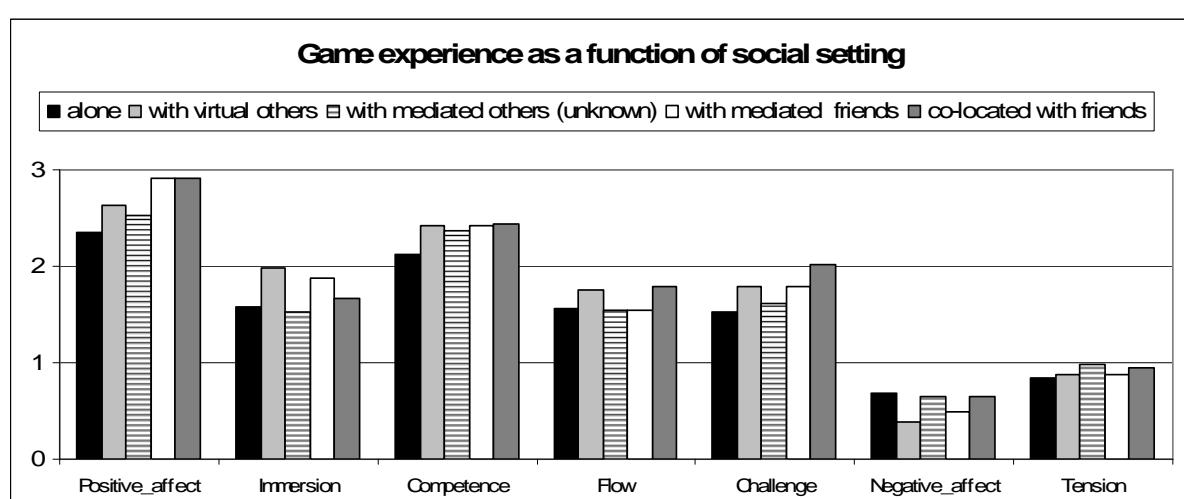
The Competence subscale differed significantly between categories,  $F(4, 348)= 2.96$ ,  $p=.02$ . Contrast analyses showed that playing alone resulted in significantly, or marginally significant lower scores than all other types of social settings, where there was some (virtual, mediated, or co-located) social identity involved in the game.

No significant differences emerged on the Flow subscale.

Differences did emerge on the Challenge subscale,  $F(4, 348)= 3.62$ ,  $p=.007$ . Participants playing alone reported lower challenge than those playing against virtual others (group 2), mediated friends (group 4), or co-located friends (group 5). This last group also scored significantly higher than those competing with online strangers (group 3).

Significant effects also appeared on the Negative affect subscale,  $F(4, 348)= 4.31$ ,  $p=.002$ . Negative affect of those playing against virtual others was lower than of those playing alone (group 1), with online strangers (group 3) or with co-located friends (group 5). We have no explanation as to why this effect occurred.

Lastly, the Tension subscale showed no significant effects of social setting.



**Figure 3.** Game experience on GEQ subscales as a function of social setting

## 4.2. In-game Questionnaire (iGEQ)

The in-game experience questionnaire (iGEQ) has the exact same component structure as the core questionnaire, but only consists of two of the five or six items for every component. We repeated some of the analyses reported above and compared them, to check whether the sensitivity of this shortened version of the GEQ is still sufficient. Although, logically, the standard deviations are somewhat bigger than for the sub scales of the core questionnaire, the results are very similar.

### Gender differences

Female players report significantly lower scores ( $p < .001$ ) on every single subscale of the iGEQ, except for Negative Affect, where females score higher, and Tension, where there is no significant difference. Results are reported in Table 4.

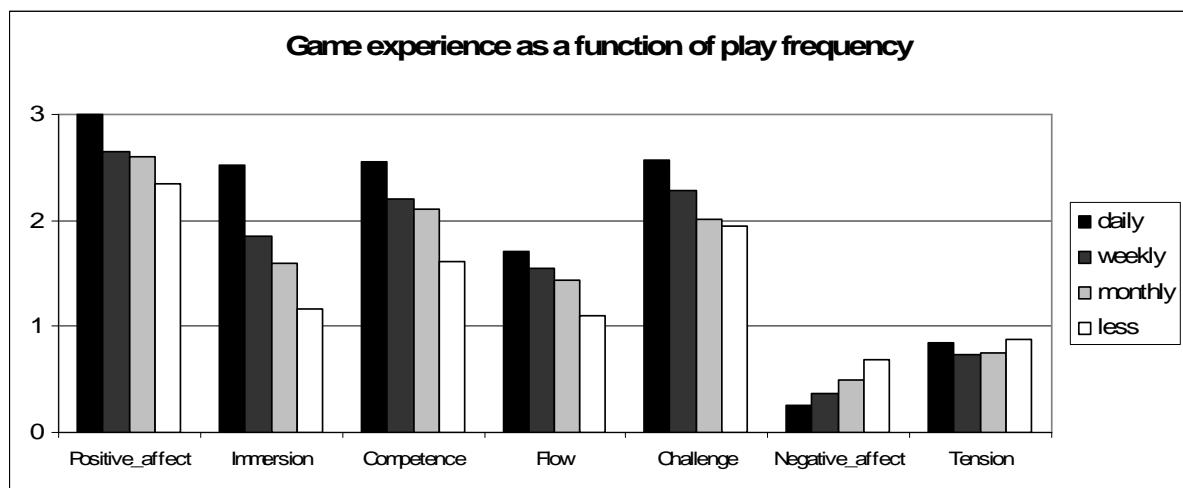
**Table 4.** Gender differences on iGEQ subscales

|        | Positive affect | Immersion   | Competence  | Flow        | Challenge   | Negative affect | Tension    |
|--------|-----------------|-------------|-------------|-------------|-------------|-----------------|------------|
| Female | 2.39 (.75)      | 1.08 (1.15) | 1.74 (.99)  | 1.20 (1.04) | 1.80 (.96)  | 0.64 (.72)      | 0.68 (.79) |
| Male   | 2.77 (.78)      | 2.17 (1.32) | 2.30 (1.32) | 1.64 (1.09) | 2.44 (1.00) | 0.35 (.64)      | 0.83 (.90) |
| Total  | 2.65 (.79)      | 1.81 (1.37) | 2.12 (1.00) | 1.49 (1.09) | 2.24 (1.03) | 0.44 (.68)      | 0.78 (.87) |

Note: N females = 120, N males = 254.

### Frequent vs. infrequent gamers

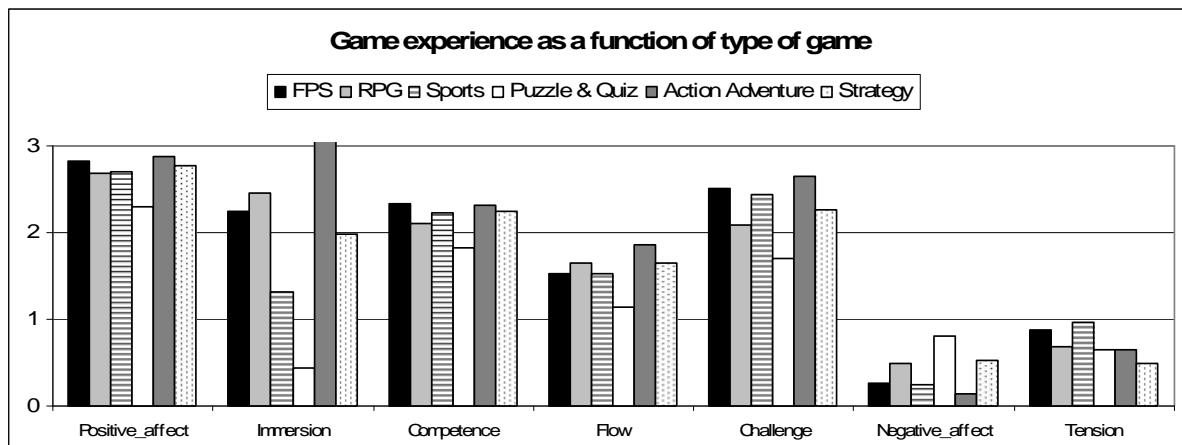
As with the core GEQ, game experience was explored as a function of how often participants played the game they reported on. Again, Positive Affect, Immersion, Competence, Flow, and Challenge all increased with play frequency, as is indicated in Figure 4. Negative Affect showed an inverted relation with play frequency. Analyses of variance showed that these differences were significant (all  $p < .01$ ). Reported tension was not related to play frequency.



**Figure 4.** Game experience on iGEQ subscales as a function of play frequency (this particular game)

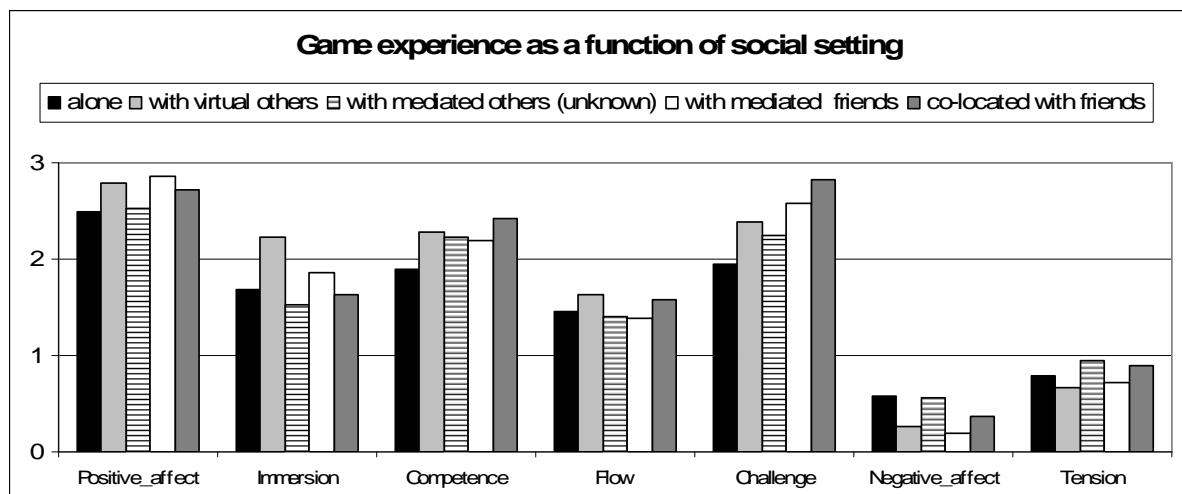
### Type of game

We subsequently explored whether game experience dimensions varied as a function of the type of game participants played. For most game experience components, analyses of variance yielded significant differences (all  $p < .05$ ) for Type of game played, identical to results on the GEQ. Scores for each game type on the game experience dimensions, are visualised in Figure 5.

**Figure 5.** Game experience on iGEQ subscales as a function of type of game

### Social setting

As with GEQ scores, iGEQ scores were compared for various social settings. The scores are visualised in Figure 6. Again response patterns were very similar to those reported for the GEQ scores. All subscales except Flow and Tension showed significant differences between social settings.

**Figure 6.** Game experience on iGEQ subscales as a function of social setting

### 4.3. Post-game Questionnaire (PGQ)

The basic descriptives for the post-game module are reported in Table 5. The scores for Positive experience are moderate, the scores on all remaining scales are low. Correlations between the subscales are modest, see Table 6.

**Table 5.** Descriptive statistics for Post-game subscales.

|                     | Mean | SD   | min | max |
|---------------------|------|------|-----|-----|
| Positive experience | 1.58 | 1.04 | 0.0 | 4.0 |
| Negative experience | .51  | .62  | 0.0 | 3.3 |
| Tiredness           | .40  | .70  | 0.0 | 4.0 |
| Return to reality   | .34  | .60  | 0.0 | 4.0 |

Note: scales range from 0 to 4

**Table 6.** Bivariate correlations between Post-game subscales.

|                     | Positive experience | Negative experience | Tiredness | Return to reality |
|---------------------|---------------------|---------------------|-----------|-------------------|
| Positive experience |                     | -.248*              | .065      | .339*             |
| Negative experience | -.248*              |                     | .338*     | .198*             |
| Tiredness           | .065                | .338*               |           | .331*             |
| Return to reality   | .339*               | .198*               | .331*     |                   |

Note: N = 380; \* = p< .001

Bivariate correlations were explored between the subscales of the Core GEQ, measuring experience while gaming, and the PGQ, measuring the players' experience after having stopped gaming. The results are reported in Table 7. A number of interesting findings emerged. For one, it is clear that, although both questionnaires were employed after the game, subscales of the two lists do measure different experiences. This indicates that participants are indeed able to discern between the two phases. Second, a positive post-game experience correlates stronger with in-game experiences of competence and immersion (.64 and .61) than with positive affect (.51). It also shows high correlations with in-game experiences of flow (.49) and challenge (.52). Lastly, results indicated that returning to reality after gaming is harder after having experienced high levels of flow (.46) and immersion (.41), which is in line with theoretical readings on these psychological concepts.

**Table 7.** Bivariate correlation between Core GEQ and post-game (PGQ) subscales.

|                      | Positive affect | Immersion | Competence | Flow   | Challenge | Negative affect | Tension |
|----------------------|-----------------|-----------|------------|--------|-----------|-----------------|---------|
| Negative experience  | -.40***         | -.25***   | -.23***    | -.06   | -.20***   | .61***          | .25***  |
| Positive experience  | .51***          | .61***    | .64***     | .49*** | .52***    | -.30***         | .17***  |
| Tiredness            | -.14**          | .05       | .01        | .22*** | .16**     | .20***          | .37***  |
| Returning to reality | .10             | .41***    | .22***     | .46*** | .26***    | .00             | .17***  |

Note: N = 375; \* p< .05, \*\*p<.01, \*\*\*p<.001

### Gender differences

Several significant gender differences were found. They are reported in Table 8. In line with the findings on the Core GEQ, Female participants scored lower on Positive experience and Return to reality, and higher on Negative experience. There was no significant difference on the Tiredness scale.

**Table 8.** Gender differences on Post-game subscales

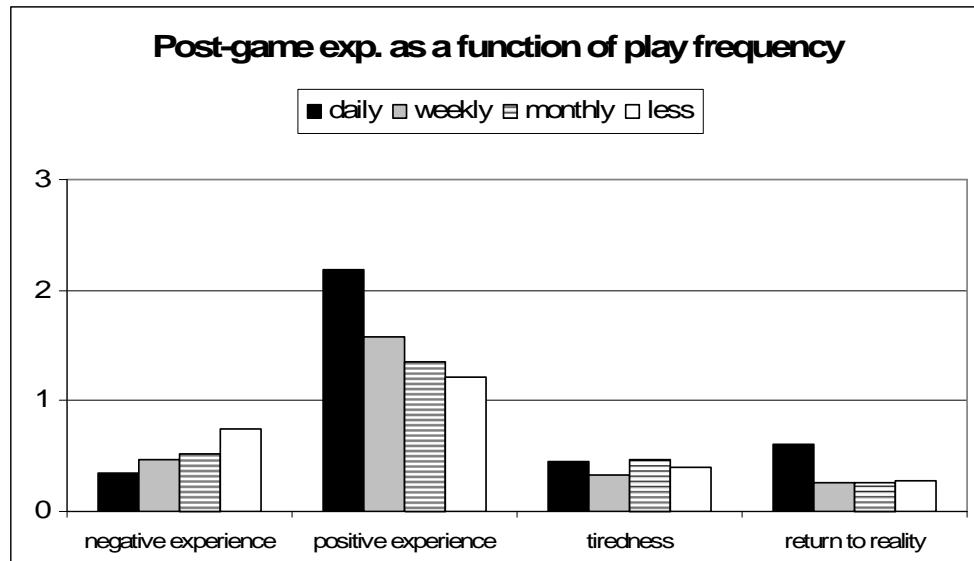
|        | Positive experience | Negative experience | Tiredness  | Return to reality |
|--------|---------------------|---------------------|------------|-------------------|
| Female | 1.14 (.89)          | 0.66 (.62)          | 0.36 (.70) | 0.16 (.33)        |
| Male   | 1.79 (1.04)         | 0.66 (.62)          | 0.41 (.70) | 0.41 (.63)        |
| Total  | 1.58 (1.04)         | 0.51 (.62)          | 0.40 (.70) | 0.33 (.58)        |

Note: N females = 120, N males = 254.

### Frequent vs. infrequent gamers

Post-game experience was explored as a function of how often participants played the game they reported on. In line with expectations, positive experience increased with play frequency, F(3, 371)=

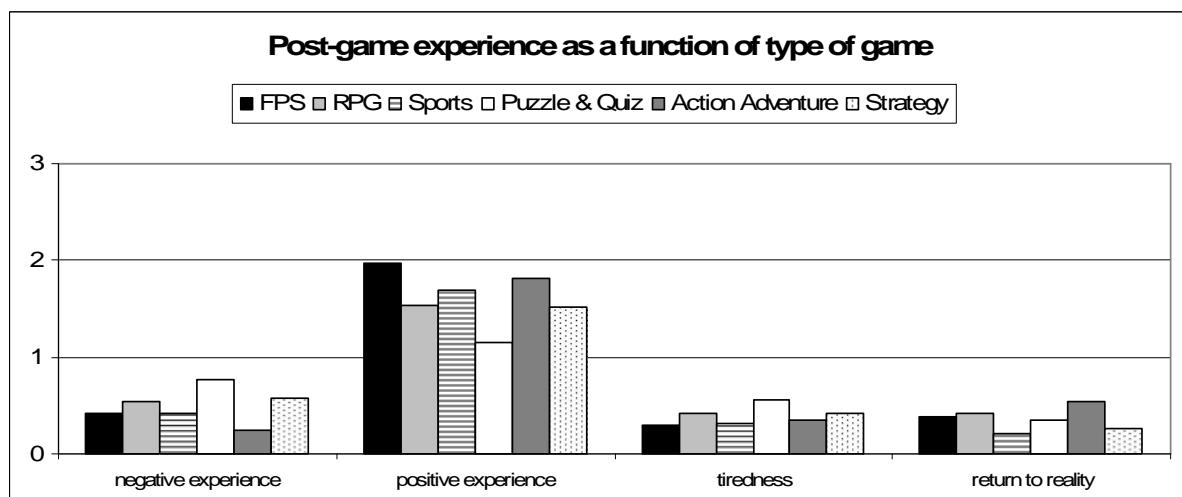
13.90,  $p<.001$ , while negative experience decreased,  $F(3, 371)= 5.68$ ,  $p=.001$ . Returning to reality was hardest for frequent players,  $F(3, 371)= 7.10$ ,  $p<.001$ . The tiredness scale did not show significant effects,  $F<1$ , NS.



**Figure 7.** PGQ subscales for different play frequencies

#### Type of game

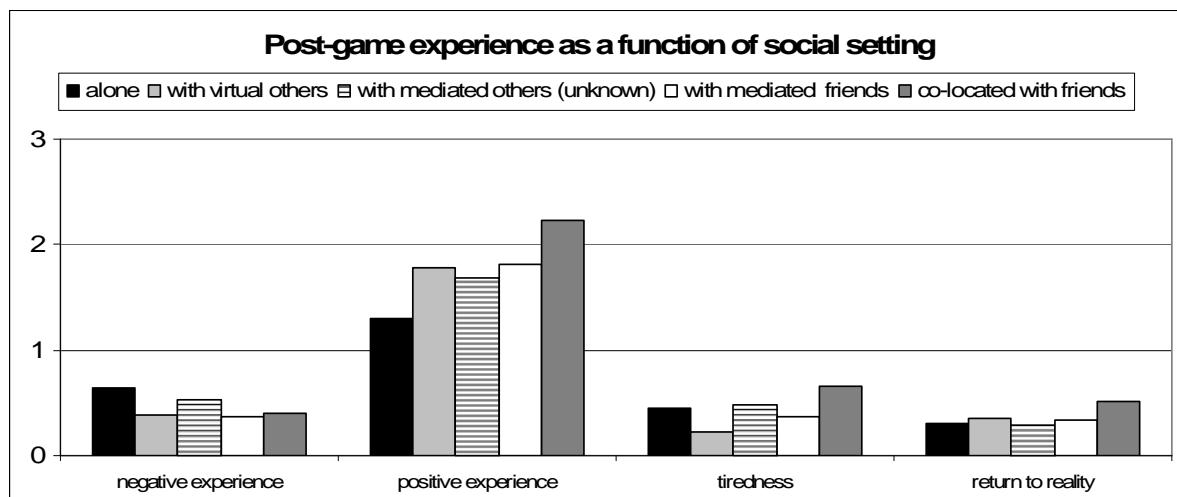
PGQ subscales were explored for different types of games (see Figure 8). Negative experience after playing differed significantly between games,  $F(5, 299)= 3.93$ ,  $p=.002$ . The most negative experiences (though still very low) were reported for puzzles and quizzes, least negative for Action adventure games. Positive experience,  $F(5, 299)= 4.62$ ,  $p<.001$ , was highest for FPS, lowest for puzzles. The remaining two scales did not show significant differences between games.



**Figure 8.** PGQ subscales for different types of games

#### Social setting

PGQ subscales also showed significant differences between different social settings (Figure 9). Players who had played alone reported the most negative experience,  $F(4, 346)= 3.08$ ,  $p=.016$ . Players who played with co-located other(s) reported the highest positive experience,  $F(4, 346)= 6.70$ ,  $p<.001$ .

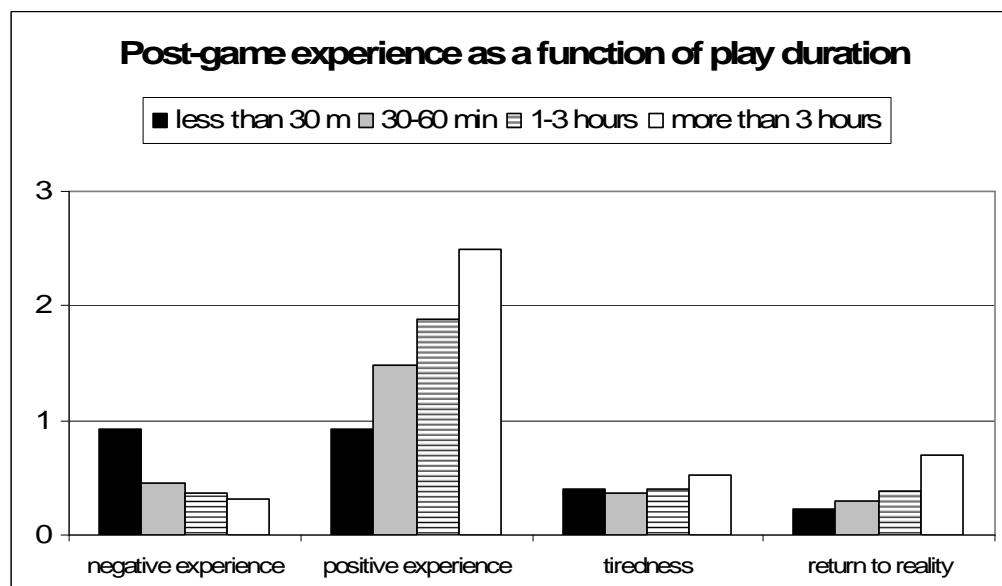


**Figure 9.** PGQ subscales for different social settings

### Play duration

Lastly, post-game experiences were explored as a function of play duration (how long participants had gamed before filling in the questionnaire, Figure 10). Perhaps surprisingly, Tiredness did not show significant differences between groups. Perhaps this indicates that games do not cause fatigue, or that the tiredness scale is not sensitive enough. Note: This scale did not result in any significant differences on the ANOVAs performed; the scale did show a few modest but significant correlations with other scales.

Furthermore, positive post-game experience was higher for participants who had played longer,  $F(3, 371) = 24.67$ ,  $p < .001$ , and their scores were lower on negative post-game experience,  $F(3, 371) = 16.48$ ,  $p < .001$ . Returning to reality was harder, the longer participants had played,  $F(3, 371) = 4.24$ ,  $p = .006$ .



**Figure 10.** PGQ subscales as a function of play duration

### 4.4. Social Presence in Gaming Questionnaire (SPGQ)

The Social Presence module consists of three subscales. Empathy and Negative feelings measure components of psychological involvement, the third scale measures Behavioural Involvement. These scales have only been filled in by participants who played against a social entity (virtual, mediated, or co-located,  $N=185$ ). Basic descriptives are given in Table 9.

**Table 9.** Descriptive statistics for Social Presence subscales

|                   | <b>Mean</b> | <b>SD</b> | <b>min</b> | <b>max</b> |
|-------------------|-------------|-----------|------------|------------|
| Empathy           | 1.41        | 1.03      | .00        | 4.00       |
| Negative feelings | .88         | .73       | .00        | 3.20       |
| Behavioural       | 2.12        | .94       | .00        | 4.00       |

Note: scales range from 0 to 5.

There were significant correlations between the subscales. The correlation between Empathy and Negative feelings was strongest,  $r= 0.45$ ,  $p<.001$ . Note that this correlation is positive, indicating that participants reporting more positive feelings towards their co-players also reported more negative feelings. This illustrates that both measures are indicators of psychological involvement: the more 'socially present' the other person is, the stronger their mutual influence on each other's feelings, both positively and negatively toned. The correlations of these scales with behavioural involvement were fairly low:  $r=.20$  for empathy, and  $r=.28$  for negative feelings.

Bivariate correlations were explored between the subscales of the Core GEQ and social presence subscales. The results are reported in Table 10. Overall, correlations were modest, but still a number of interesting findings emerged. All the competence and challenge subscales of the GEQ showed the strongest relationships with all three subscales of social presence. Game experiences of competence and challenge are associated with higher psychological involvement between co-players, both as regards positive and negative feelings, and higher behavioural involvement. In addition, negative feelings towards the co-player (jealousy, revenge, influencing each others' moods) were most strongly related to reported tension, whereas positive feelings towards the co-player (empathy) was correlated highest with reported positive affect and immersion.

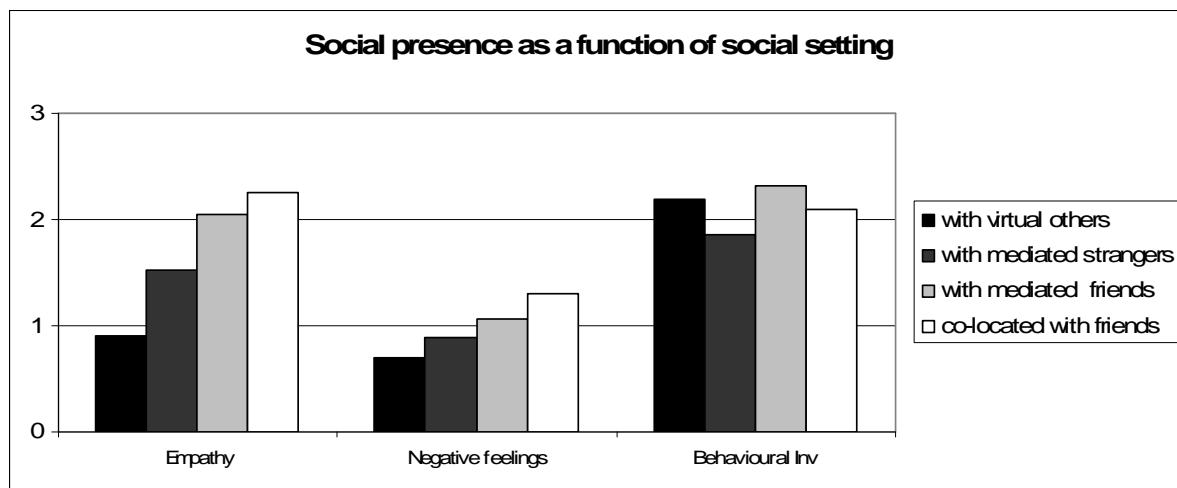
**Table 10.** Bivariate correlation between Core GEQ and Social presence subscales.

|                         | Positive affect | Immersion | Competence | Flow   | Challenge | Negative affect | Tension |
|-------------------------|-----------------|-----------|------------|--------|-----------|-----------------|---------|
| Empathy                 | .31***          | .28***    | .23***     | .10    | .22***    | .00             | .10     |
| Negative feelings       | .13*            | .16*      | .22***     | .24*** | .30***    | .14*            | .36***  |
| Behavioural involvement | .16*            | .10       | .26***     | .18**  | .27***    | -.03            | .14*    |

Note: N = 224; \*  $p < .05$ , \*\* $p < .01$ , \*\*\* $p < .001$

### Social setting

The social presence subscales were explored for various social settings. The results are shown in Figure 11. Significant differences were found for both psychological involvement scales. Empathy increased with increasing social presence and relatedness,  $F(3, 181)= 21.83$ ,  $p<.001$ . Contrast analysis showed that scores increased significantly ( $p<.001$ ) with each subsequent category. Although modest for all groups, negative feelings also differed significantly between settings,  $F(3,181)= 5.38$ ,  $p=.001$ . The contrast between the 1<sup>st</sup> and 2<sup>nd</sup> category (virtual vs. mediated strangers) did not reach significance, but all remaining contrasts did, indicating higher scores for negative feelings with increasing social presence and relatedness. Behavioural involvement was equal for all social settings: even for settings with low social presence and relatedness, players' behaviour is influenced by the other social entity's behaviour. This makes a lot of sense in view of the present application area: in a game, the player's and opponent's actions are interdependent and make up the very core of the activity.



**Figure 11.** Social presence subscales for different social settings

#### 4.5. Conclusion

The present chapter presented statistical explorations to check sensitivity and validity of the newly developed self-report measures. For each scale, average scores and response ranges were calculated, and bivariate correlations reported. Subsequently, response patterns were explored for different types of gamers (male vs. female, and frequent vs. infrequent gamers), different game types, and social settings.

Average scores on subscales of the core GEQ were moderate (as planned) and ranged the full scale. Exceptions to this are negative affect and tension, for which the average scores were somewhat low. The highest scores appeared on positive affect and immersion. Analyses of bivariate correlations indicated that, although related, the seven subscales do measure different components of experience. Discriminant validity was further demonstrated in subsequent analyses of variance (ANOVAs), where components each showed unique response patterns for gamer characteristics, game types and social settings.

Male players reported more game enjoyment overall than female players, as indicated by higher scores for positive affect, competence, flow, immersion, and challenge, and lower scores for negative affect. Similarly, frequent players reported higher enjoyment than infrequent players, demonstrated in almost linear relationships between scores on GEQ scales and play frequency. Significant differences were also found for game type, based on post-hoc categorisation. Playing puzzles and quizzes was generally reported as less involving and enjoying than other types of games. Action adventure games scored particularly high on immersion, followed by FPS and role playing games. Lastly, analyses of various social settings showed that with increasing social presence and relatedness to others both positive affect and challenge increased. These findings are in line with a priori expectations and earlier findings and thus corroborate validity of the measures.

Although standard deviations are somewhat bigger for the iGEQ than for GEQ subscales, statistical analyses rendered very similar results. In spite of its conciseness, sensitivity and validity of this measure are still highly satisfactory.

For PGQ scales the average score for the positive experience subscale was moderate, the scores on all remaining scales were low. Bivariate correlations between GEQ and PGQ subscales clearly indicated that both scales measure different experiences. High scores on positive post-game experience correlated with in-game positive affect, but even stronger with competence and immersion. Feeling good afterwards thus mainly depends on whether the experience was rich and whether players felt competent in what they were doing. Flow and immersion experienced during the game were also related to experiencing the game a positive experience afterwards. Returning to reality after gaming was harder after having experienced high levels of flow and immersion.

Female participants scored lower on positive experience and returning to reality, and higher on negative experience, which corroborates the gender differences found on the GEQ. Similarly, positive experience and difficulty with returning to reality increased with play frequency and play duration, while negative experience decreased, in line with GEQ findings of higher enjoyment and involvement.

Experience after playing was most positive after FPSs, most negative after puzzles and quizzes. Furthermore, results indicated that players who had played alone reported the most negative experiences, whereas those playing with co-located other(s) reported the highest positive experiences.

SPGQ subscales showed low to moderate average scores. The two psychological involvement scales, measuring the experience and contagion of positive (empathy) and negative feelings, were positively correlated, illustrating that both measures are indicators of social presence: the more 'socially present' the other person is, the stronger their mutual influence on each other's feelings, both positively and negatively toned.

In line with expectations, psychological involvement (both empathy and negative feelings) increased with social presence and relatedness. Behavioural involvement was equal for all social settings: even for settings with low social presence and relatedness, players' behaviour is influenced by the other social entity's behaviour. This result may well be unique to the present application area: in a game, the player's and opponent's actions are interdependent and make up the very core of the activity. This is not necessarily the case in other social presence technology applications.

The SPGQ experiences – psychological involvement between co-players, both as regards positive and negative feelings, and higher behavioural involvement – were associated with higher GEQ experiences of competence and challenge. In addition, negative feelings towards the co-player (jealousy, revenge, influencing each others' moods) were most strongly related to reported tension, whereas positive feelings towards the co-player (empathy) was correlated highest with reported positive affect and immersion.

Summing up these findings, statistical analyses firmly demonstrated that the GEQ and additional modules were sensitive enough to pick up differences between gamers, game types, play characteristics, and social context of play. Moreover, each subscale's unique value (discriminant validity) has been demonstrated by different response patterns to variations in these background variables.

## 5. Discussion and conclusion

### 5.1. Questionnaire development

The general aim in developing the GEQ was to construct a reliable, valid and sensitive, comprehensive measure of game experience. The theoretical work performed by all partners in the FUGA project (Deliverable 2.1) provided a firm and broad basis for the development of the Game Experience Questionnaire. Additional relevant scientific literature was reviewed for any conceptualisations that might complement this work and for existing scales that might serve as a starting point or inspiration for item formulation. Furthermore, empirical data was gathered during focus group interviews with different types of gamers, discussed in Chapter 2. These interviews served as a test for comparing scientific conceptualisations and lay descriptions of first-hand experiences. They were also used as a reference guide on choice of wording in the item formulation phase.

The Game Experience Questionnaire was developed with a modular structure, consisting of:

4. The core questionnaire (GEQ). This is the heart of the Game Experience Questionnaire, probing multiple components of players' experience while gaming.
5. The post-game questionnaire (PGQ), probing gamers' experience after the gaming session and any after effects.
6. The social presence module (SPGQ), probing gamers' experience of and involvement with their co-player(s).

These three lists are to be administered after the gaming session has ended. Additionally, a short in-game version of the GEQ was developed, the iGEQ, for probing in-game experience multiple times during a gaming session.

A large scale survey was performed to test the long list (>100 items) and explore the factor structures of the questionnaires. Factor analyses provided structures for the scales that made good sense in the light of theoretical considerations, with subscales that were all easy to interpret. Subsequent reliability tests resulted in the construction of reliable subscales with satisfactory to high internal consistencies. Statistical analyses of differences between gamers (gender, play frequency), all demonstrated the sensitivity of the scales, as well as each subscale's unique value (discriminant validity).

The full questionnaire is listed in Appendixes 1-4. Scoring guidelines for all scales are listed in Appendix 5.

### 5.2. Assessment of measurement criteria for the GEQ

In Chapter one, a list of criteria for (self-report) measures of game experience was discussed. According to this overview, the GEQ should preferably be tested on – and meet – criteria for reliability, validity, sensitivity, robustness, non-intrusiveness, and convenience.

For the core GEQ, iGEQ, PGQ, and SPGQ, we reported on the internal consistency of all subscales. Results showed that all measures are reliable, with satisfactory to excellent Cronbach's alpha values. The test-retest reliability of the entire battery of measures developed by FUGA (including the GEQ) will be established in WP6. Some limitations were noted for the PGQ, for which item variabilities were suboptimal. Although even these scales performed well on the remaining criteria below (except for the tiredness scale), more research is needed before this scale's final structure can be established.

Face validity and content validity were established in the construction phase of the GEQ, through the explicit use of the theoretical framework developed under FUGA, as well as recent conceptualisations in scientific literature, and the exploration of lay conceptualisations collected through focus group interviews. Predictive validity will be established through a series of studies performed under WP7. Convergent validity is in fact at the heart of the FUGA multi-measure approach, as we are comparing and correlating the outcomes of different measurement methods throughout the project lifetime. For the GEQ and its additional modules, discriminant validity was established internally for the various subscales, as different scales showed different response patterns to variations in player type, game content, and setting.

In Chapter 4 of the present deliverable, we analysed the sensitivity of the GEQ in relation to background variables such as, gender, play frequency, type of game, and social setting. The subscales of the GEQ and modules were able to significantly distinguish between different levels of these background variables. The single exception to this is the tiredness subscale of the PGQ.

The GEQ has explicitly been developed to probe experience ‘sec’, i.e., without reference to specific game or interface characteristics. This will aid the tool’s robustness, i.e., the wide applicability of the questionnaire, as was demonstrated in the large scale survey, which assessed game experience of players of highly diverse backgrounds, playing a multitude of games, on various platforms in different social and physical contexts.

The GEQ, PGQ, and SPGQ, being post-test self-report measures, can be assumed to be non-intrusive. The intrusiveness of the in-game version of the GEQ, the iGEQ, is minimised through its limited number of items. Moreover, researchers can decide to discard more items from this list when they are targeting specific components of experience. Each component is assessed with only two items, which can be administered extremely quickly.

Lastly, as a low-cost paper-based measure, the GEQ can be regarded as highly convenient for researchers to use. It is easy to learn and easy to administer. The fact that the questionnaire will be translated into English, Finnish, Swedish and German, and be made available to all interested parties only adds to this effect. More importantly, convenience for research participants was also taken into account, by formulating brief items using simple, natural, and common language, and the consistent use of one type of answering scale.

### **5.3. Further steps in the development of the GEQ**

The questionnaire has been provided to project partners for translation to Finnish, Swedish, and German. An English version of the GEQ is currently being tested by TU/e.

Translation instructions for partners were given as follows:

1. Two translators are needed to translate and back-translate the questionnaires to the new language.
2. Translator 1 translates the English items into the new language. The aim is to stay as close as possible to the meaning of the original English questionnaire. However, the translated item should feel natural and logical. The translator should be proficient and fluent in English and a native speaker in the new language.
3. A second translator translates the new items back into the English language. Translator 2 is proficient and fluent in the new language and an English native speaker.
4. The new English translation is compared to the original English version. The meaning, direction and intensity of the items should be identical. Where differences occur, the two translators discuss the discrepancy and decide on a translation that unambiguously probes what the original English item is meant to probe.
5. Please do not forget to also carefully translate the instructions and answering categories.
6. For your convenience, a second file (excel) is distributed, which contains the English items structured according to the components they belong to, as well as the Dutch versions of the items.

After the translated scale has been used for the first time, statistical analysis will be performed to confirm factor structure and internal consistencies of the scales. The extra items provided for translations of the GEQ scale allow for some flexibility in adopting or discarding single items from subscales in order to construct reliable scales.

As discussed above, the PGQ structure will be further explored in future empirical work. It should be noted that this scale is not crucial to experimentations within the FUGA project, as it is focused on developing continuous measurements for in-game experience. Also, in spite of its restrictions, three of its subscales did demonstrate reliability, validity, and sensitivity in their present form.

Empirical studies planned in the FUGA (see Deliverable 3.1) will employ the GEQ. This will provide additional data for further validating the questionnaire.

## References

- Bartle, R.A. (1996). *Hearts, clubs, diamonds and spades: players who suit MUDs.* <http://www.mud.co.uk/richard/hcds.html>.
- Biocca, F., Harms, C., Burgoon, J., (2003). Toward a more robust theory and measure of social presence: Review and suggested criteria. *Presence*, 12, 456–480.
- Biocca, F., Harms, C., Gregg, J. (2001). *The Networked Minds Measure of Social Presence: Pilot Test of the Factor Structure and Concurrent Validity*. E. Lansing, MI: Media Interface and Network Design (M.I.N.D.) Lab.
- Brown, E. & Cairns, P. (2004). A grounded investigation of game immersion. *ACM CHI 2004*, 1297-1300.
- Costello, A. B. & Osborne, J. (2005). Best practices in exploratory factor analysis: four recommendations for getting the most from your analysis. *Practical Assessment Research & Evaluation*, 10 (7). Available online: <http://pareonline.net/getvn.asp?v=10&n=7>
- Cronbach, L. J. (1990) Essentials of psychological testing. 5<sup>th</sup> edition. New York: Harper Collins.
- Deci, E. L., & Ryan, R. M. (1985). *Intrinsic motivation and self determination in human behavior*. New York: Plenum.
- Deci, E. L., & Ryan, R. M. (2000). The “what” and “why” of goal pursuits: Human needs and the self-determination of behavior. *Psychological Inquiry*, 11, 227–268.
- Ermi, L. & Mäyrä, F. (2005). Fundamental components of the gameplay experience: Analysing immersion. In: S. de Castell & J. Jenson (eds.), *Changing Views: Worlds in Play*. Selected papers of the 2005 Digital Games Research Association’s Second International Conference.
- Fabrigar, L. R., Wegener, D. T., MacCallum, R. C., & Strahan, E. J. (1999). Evaluating the use of exploratory factor analysis in psychological research. *Psychological Methods*, 4, 272-299.
- Intrinsic Motivation Inventory, <http://www.psych.rochester.edu/SDT/measures/word/IMIfull.doc> (retrieved on 01-10-2003).
- Lessiter, J., Freeman, J., Keogh, E., & Davidoff, J. (2001). A cross-media presence questionnaire: The ITC-Sense of Presence Inventory. *Presence*, 10, 282-297.
- Lunt, P. & Livingstone, S. (1996). Rethinking the focus group in media and communications research. *Journal of Communication*, 46(2), 79-98.
- Merton, R.K. (1987). The focused interview and focus groups: continuities and discontinuities. *Public Opinion Quarterly*, 51, 550-556.
- Miller, L. et al. (1996). Female participants' Preferences in Software Design: Insights from a Focus Group. *Interpersonal Computing and Technology*, 4(2), 27-36.
- Morgan, D. (1998). *The focus group guidebook. Focus group kit 1*. Thousand Oaks, CA: Sage.
- Ryan, R. M., Rigby, C. S., & Przybylski, A. (2006). The Motivational Pull of Video Games: A Self-Determination Theory Approach. *Motivation and Emotion*, 30, 347–363.
- Ryan, R. M., Mims, V., & Koestner, R. (1983). Relation of reward contingency and interpersonal context to intrinsic motivation: A Review and test using cognitive evaluation theory. *Journal of Personality and Social Psychology*, 45, 736–750.
- Sweetser, P & Wyeth, P. (2005). GameFlow: A model for evaluating player enjoyment in games. *ACM Computers in Entertainment*, 3 (3), 1-24.
- Tabachnick, B. G., & Fidell, L. S. (2001). *Using Multivariate Statistics*. Boston: Allyn and Bacon.
- Watson, D., Clark, L. A., & Tellegen, A. (1988). Development and validation of brief measures of Positive and Negative affect: the PANAS Scales. *Journal of Personality and Social Psychology*, 47, 1063–1070.
- Yee, N. (2002). *Facets: 5 motivation factors for why people play MMORPG's.* <http://www.nickyee.com/facets/home.html>.

## Appendix 1: Game Experience Questionnaire (GEQ)

Please indicate how you felt while playing the game for each of the items,  
on the following scale:

| not at all | slightly | moderately | fairly | extremely |
|------------|----------|------------|--------|-----------|
| 0          | 1        | 2          | 3      | 4         |
| < >        | < >      | < >        | < >    | < >       |

- 1 I felt content
- 2 I felt skilful
- 3 I was interested in the game's story
- 4 I could laugh about it
- 5 I felt completely absorbed
- 6 I felt happy
- 7 I felt tense
- 8 I felt that I was learning
- 9 I felt restless
- 10 I thought about other things
- 11 I found it tiresome
- 12 I felt strong
- 13 I thought it was hard
- 14 It was aesthetically pleasing
- 15 I forgot everything around me
- 16 I felt good
- 17 I was good at it
- 18 I felt bored
- 19 I felt successful
- 20 I felt imaginative
- 21 I felt that I could explore things
- 22 I enjoyed it
- 23 I was fast at reaching the game's targets
- 24 I felt annoyed
- 25 I was distracted
- 26 I felt stimulated
- 27 I felt irritable
- 28 I lost track of time
- 29 I felt challenged
- 30 I found it impressive
- 31 I was deeply concentrated in the game
- 32 I felt frustrated

- 33 It felt like a rich experience
- 34 I lost connection with the outside world
- 35 I was bored by the story
- 36 I had to put a lot of effort into it
- 37 I felt time pressure
- 38 It gave me a bad mood
- 39 I felt pressured
- 40 I was fully occupied with the game
- 41 I thought it was fun
- 42 I felt competent

## Appendix 2: In-game Questionnaire (iGEQ)

Please indicate how you felt while playing the game for each of the items,  
on the following scale:

|            |          |            |        |           |
|------------|----------|------------|--------|-----------|
| not at all | slightly | moderately | fairly | extremely |
| 0          | 1        | 2          | 3      | 4         |
| < >        | < >      | < >        | < >    | < >       |

- |  |               |
|--|---------------|
| 1 I was interested in the game's story | GEQ Core – 3  |
| 2 I felt successful                    | GEQ Core – 19 |
| 3 I felt bored                         | GEQ Core – 18 |
| 4 I found it impressive                | GEQ Core – 30 |
| 5 I forgot everything around me        | GEQ Core – 15 |
| 6 I felt frustrated                    | GEQ Core – 32 |
| 7 I found it tiresome                  | GEQ Core – 11 |
| 8 I felt irritable                     | GEQ Core – 27 |
| 9 I felt skilful                       | GEQ Core – 2  |
| 10 I felt completely absorbed          | GEQ Core – 5  |
| 11 I felt content                      | GEQ Core – 1  |
| 12 I felt challenged                   | GEQ Core – 29 |
| 13 I felt stimulated                   | GEQ Core – 26 |
| 14 I felt good                         | GEQ Core – 16 |

### Appendix 3: Post-game experience questionnaire (PGQ)

Please indicate how you felt after you finished playing the game for each of the items, on the following scale:

| not at all | slightly | moderately | fairly | Extremely |
|------------|----------|------------|--------|-----------|
| 0          | 1        | 2          | 3      | 4         |
| < >        | < >      | < >        | < >    | < >       |

- 1 I felt revived
- 2 I felt bad
- 3 I found it hard to get back to reality
- 4 I felt guilty
- 5 It felt like a victory
- 6 I found it a waste of time
- 7 I felt energised
- 8 I felt satisfied
- 9 I felt disoriented
- 10 I felt exhausted
- 11 I felt that I could have done more useful things
- 12 I felt powerful
- 13 I felt weary
- 14 I felt regret
- 15 I felt ashamed
- 16 I felt proud
- 17 I had a sense that I had returned from a journey

## Appendix 4: Social Presence Gaming Questionnaire (SPGQ)

Please indicate how you felt while playing the game for each of the items,  
on the following scale:

|            |          |            |        |           |
|------------|----------|------------|--------|-----------|
| not at all | slightly | moderately | fairly | extremely |
| 0          | 1        | 2          | 3      | 4         |
| < >        | < >      | < >        | < >    | < >       |

- 1 I empathized with the other(s)
- 2 My actions depended on the other(s) actions
- 3 The other's actions were dependent on my actions
- 4 I felt connected to the other(s)
- 5 The other(s) paid close attention to me
- 6 I paid close attention to the other(s)
- 7 I felt jealous about the other(s)
- 8 I found it enjoyable to be with the other(s)
- 9 When I was happy, the other(s) was(were) happy
- 10 When the other(s) was(were) happy, I was happy
- 11 I influenced the mood of the other(s)
- 12 I was influenced by the other(s) moods
- 13 I admired the other(s)
- 14 What the other(s) did affected what I did
- 15 What I did affected what the other(s) did
- 16 I felt revengeful
- 17 I felt schadenfreude (malicious delight)

## Appendix 5: Scoring guidelines

### Scoring guidelines GEQ

The Core GEQ Module consists of seven components, the items for each are listed below.

Most components have 5 items (except the SII which has 6). We have added one extra item for each of these, to create some flexibility should a translated item not work. Please also translate and use these items in your first studies, until the translated scales have been tested.

Component scores are computed as the average value of its items.

**Competence:** Items 2, 12, 17, 19, and 23; 42 is a spare item for translation..

**Sensory and Imaginative Immersion:** Items 3, 14, 20, 21, 30, and 33.

**Flow:** Items 5, 15, 28, 31, and 34; 40 is a spare item for translation.

**Tension:** Items 7, 9, 24, 27, and 32; 39 is a spare item for translation.

**Challenge:** Items 8, 13, 26, 29, and 36; 37 is a spare item for translation.

**Negative affect:** Items 10, 11, 18, 25, and 35; 38 is a spare item for translation.

**Positive affect:** Items 1, 4, 6, 16, and 22; 41 is a spare item for translation.

### Scoring guidelines iGEQ

The In-game questionnaire consists of seven components, identical to the core Module. However, only two items are used for every component. The items for each are listed below.

Component scores are computed as the average value of its items.

**Competence:** Items 2 and 9.

**Sensory and Imaginative Immersion:** Items 1 and 4.

**Flow:** Items 5 and 10.

**Tension:** Items 6 and 8.

**Challenge:** Items 12 and 13.

**Negative affect:** Items 3 and 7.

**Positive affect:** Items 11 and 14.

### Scoring guidelines PGQ

The post-game Module consists of four components, the items for each are listed below.

Component scores are computed as the average value of its items.

**Positive Experience:** Items 1, 5, 7, 8, 12, 16.

**Negative experience:** Items 2, 4, 6, 11, 14, 15.

**Tiredness:** Items 10, 13.

**Returning to Reality:** Items 3, 9, 17.

### Scoring guidelines SPGQ

The Social Presence Module consists of three components, the items for each are listed below.

Component scores are computed as the average value of its items.

**Psychological Involvement – Empathy:** Items 1, 4, 8, 9, 10, and 13.

**Psychological Involvement – Negative Feelings:** Items 7, 11, 12, 16, and 17.

**Behavioural Involvement:** Items 2, 3, 5, 6, 14, and 15.

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/221157583>

# Modeling enjoyment preference from physiological responses in a car racing game

Conference Paper · August 2010

DOI: 10.1109/ITW.2010.5593337 · Source: DBLP

---

CITATIONS  
61

READS  
110

---

4 authors:



Simone Tognetti  
Empatica S.r.l.  
17 PUBLICATIONS 281 CITATIONS

[SEE PROFILE](#)



Maurizio Garbarino  
Politecnico di Milano  
9 PUBLICATIONS 248 CITATIONS

[SEE PROFILE](#)



Andrea Bonarini  
Politecnico di Milano  
232 PUBLICATIONS 2,331 CITATIONS

[SEE PROFILE](#)



Matteo Matteucci  
Politecnico di Milano  
274 PUBLICATIONS 3,249 CITATIONS

[SEE PROFILE](#)

Some of the authors of this publication are also working on these related projects:



Krog - Kinect RObot for Gaming - Polisocial [View project](#)



Incremental Dense 3D Reconstruction [View project](#)

# Modeling enjoyment preference from physiological responses in a car racing game

Simone Tognetti, Maurizio Garbarino, Andrea Bonarini, Matteo Matteucci

**Abstract**— We propose a framework to estimate player enjoyment preference from physiological signals. This can produce objective measures that could be used to adapt dynamically a game to maintain the player in an optimal status of enjoyment. We present a case study on The Open Racing Car Simulator (TORCS) video game. In particular, we focus both on the experimental protocol, which we designed with special attention to produce physiological responses related to the game experience only, and on signal analysis, which produces a simple and general model good enough to estimate player enjoyment preference in real applications.

## I. INTRODUCTION

“A good game is the one you like to play”. With this motto in her mind, the digital game designer conceives rules and structure of a game to maximize the level of enjoyment for the target audience. Realistic ambiance and characters, intelligent and reactive opponents with a human-like behavior can provide enjoyable game experience, however this cannot be assumed a priori. Some evaluation can be done by considering performance parameters, often assuming that a better performance is related to higher enjoyment of the game experience; but, different players may react differently to game features, and enjoyment has to be considered as a personal experience. According to the affective computing, we can assume that physiological response can be related to enjoyment [1], and that it can be taken as an objective measure of it. Thus, we present in this paper the application of a methodological framework for the estimate of preference among game experiences, from the physiological state of the player, in the car racing video game scenario implemented by TORCS – The Open Racing Car Simulator [2].

With the purpose of comparing and validating the proposed game scenario with respect to recent literature, we have carried out a correlation analysis between physiological data and subject preference during different variants of the game. Results show that features derived from some of the physiological signals (e.g., Galvanic Skin Response (GSR), Blood Volume Pulse (BVP) and Respiration (RESP)) have a high correlation with player reported preferences. On the other hand, as we expected from other studies in literature, only some physiological features show relevant high discriminating power. Therefore, signals such as Heart Rate (HR) or temperature are not really suitable as emotional input because of their poor correlation with user preference. Supported by these findings, we have estimated a linear

model based on physiological signals able to predict the subject enjoyment preference during the game. Our approach focuses on the differential comparison of preference between game situations and the resulting model can be used, in a future experiment, to modify at runtime the game experience accordingly to the predicted user preference to optimize user enjoyment.

The ground truth about subject preference has been evaluated by questionnaire analysis. The players are asked to express a preference between two variants of the video game. This approach of eliciting emotion is named comparative affect analysis and it was first introduced by Yannakakis and Hallam [3], [4]. Affective models are then derived using preference learning techniques [5], [6] matching user reported preferences and features from physiological signals measured during the game session. We assume that physiological responses are not task dependent since the level of physical activity required for the interaction with our game is constant during the session. As presented by Yannakakis [7], the preference model can be learned using different computational methods. In this paper, we propose a different linear model obtained with Linear Discriminant Analysis (LDA) [8], which shows performance analogous to other models presented in literature [9], but lower complexity and lower computation demand.

In the next sections, we first give an insight into the state of the art, then we introduce the experimental protocol designed for our experiments, finally we present the methods we adopted for preference modeling and the results obtained.

### A. State of the art

With the purpose of determining criteria that contribute to player satisfaction, Yannakakis and Hallam [10] proposed two techniques for modeling player satisfaction in real-time. They assume that player-opponent interaction primarily contributes to the entertainment in a computer game. Therefore, metrics based on qualitative considerations of what is enjoyable from in-game performances (e.g., time before the player loses a life) have been considered as indicators of the level of interest.

Another approach consists in modeling the entertainment by following the theoretical principles described by Malone [11], and concepts related to the Theory of Flow [12]. Qualitative factors such as challenge, fantasy and curiosity are the ones that, according to them, mostly account for player entertainment. Quantitative measures for challenge, curiosity and flow state can be derived from an empirical analysis of player responses to game mechanisms. All the

Politecnico di Milano IIT Unit, Dip. di Elettronica e Informazione, via Ponzio 34/5, 20133 Milano, Italy. E-mail: {tognetti,garbarino,bonarini,matteucci}@elet.polimi.it.

metrics evaluated in the mentioned papers are specific for a given class of games (e.g., predator/prey, racing, football), therefore a general model can not be determined with these approaches. Moreover, player responses can be affected by factors such as user experience and motion or perception skills, which can be loosely related to enjoyment, while physiological responses seem to be better indicators.

Under a different perspective, emotion has been investigated in the past by many researchers, including philosophers, psychologists, sociologists, psychophysicists, and engineers. Results from psychophysiological studies [1], [13] indicated that relationship between the stimuli presented to a person and observed physiological reactions may exist. Grounded on these findings, people working on Affective Computing aimed to design human-machine interfaces, with emotion recognition abilities, for real life applications [14], [15], [16]. Recently, this research line has been extended to video games in which experiments can be performed with a good trade-off between reality and control.

In Mandryk et al. [17], statistically significant correlation has been claimed between GSR and reported fun (from questionnaire) in adults playing video games. Other physiological signals such as jaw electromyography, electrocardiography and respiration have been analyzed, but they resulted not to be correlated to the reported enjoyment. A fuzzy model inspired by psychophysiology theory is introduced by Mandryk and Atkins [18]. They reported that high values of HR and GSR together with a smile detection from an electromyography (EMG) in jaw are correlated to high values of arousal and positive valence.

Rani et al. [19] used psychophysiological measures such as HR and GSR to discriminate anxiety level and adjusted a Pong game to respond accordingly. In the approach proposed, they handle the problem of enjoyment maximization by appropriately minimizing the anxiety level.

Tijss et al in [20] showed values of skin conductance, HR and respiration to be statistically correlated to different difficulty levels of PacMan. The correlation with enjoyment state of a player is not directly calculated but it is inferred from a questionnaire analysis. Prediction models based on physiological state (HR, GSR) have also been proposed for potential entertainment augmentation in computer games [21].

In this work, we have applied some of the techniques presented in literature for physiological signal analysis for video game enjoyment evaluation. Preference learning techniques have been applied as presented by Yannakakis [9].

## II. EXPERIMENTAL SETTING

We propose a new gaming experimental protocol, tested on a car-racing computer game, in which affective computing techniques can be applied and validated. The protocol was designed to produce an affective computing benchmark dataset, that could be used also for further developments. This dataset is composed by physiological data, questionnaire answers regarding user data (i.e., game experiences and race preferences), game logs and 2 video camera recordings. 75 volunteers (60 males and 15 females) aged from 18 to 30

years old (57 from 19-25, 18 from 26-30) took part in this study.

### A. Task Design

The cognitive task in the experiment concerns playing a video game. This makes it possible to reach a high repeatability and a high level of involvement among participants. TORCS [2] was chosen as reference game for the following reasons: it is a video game that requires the player to be sitting in front of a computer, therefore subjects experiment emotionally different situations characterized by a similar physical activity and the effects of movement artifacts on acquired data are negligible differently from what happened in [9] where the subjects had to move and jump; this game is an open source project, therefore, it has been possible to implement custom logging and AI for opponent drivers; it is easy enough, even for an inexperienced player, thus the game experience can be kept as homogeneous as possible among subjects involved in the experiment.

During a game session, each participant played 7 races versus one computer driver that is the only opponent during the race. The opponent skill has been changed among races considering that this has a high potential impact on player emotional state, and that it can be easily adapted in a real-time affective loop to maintain the enjoyment level on the player according to the general principles of game engagement proposed by Malone [11].

Three classes of game scenarios have been considered and a customized opponent driver has been implemented to match the skill of the player. It modulates its speed to keep a given distance from the human driven car. We call W (Winner) the driver that is more skilled than the player and that has the goal to keep a distance of +100 m (relative distance between the cars within the current track) from the player. C (Challenging) is the driver that is as skilled as the player and tries to keep a distance of 0 m from the player. Finally L (Loser) is the driver that is less skilled than the player and that keeps a distance of -100 m from the player. According to a priori considerations, the second variant of the race could be considered really challenging, and more interesting for the player. Race parameters such as type of track, environmental details, car model, and number of opponents have been chosen to keep the game easy to play and to make the opponent skill being the main difference among races.

### B. Experimental Protocol

Most of the choices in the experimental protocol have been made to maximize the focus of the player on the task. The environment where the experiment took place has been conceived with the purpose of isolating the player and maximizing the game immersion so that no external event could influence the subject physiological state. The setting was a small room with a computer placed on a desk. The player was sitting in front of the monitor and was interacting with the computer through standard mouse and keyboard.

No other people were in the same room and the operator monitored the experiment from an external site.

Ahead of the experiment, all participants have been asked to fill out a general questionnaire, presented in computer-based form, and used to gather information about their experience with video games, game preference, TORCS prior knowledge, and personal data such as age and handedness.

Then, Participants have been fitted with sensors to measure peripheral physiological activity as explained in Section II-C. The players were asked to wear a headphone to guarantee a deeper game involvement through race sounds. After this setup phase, players were left alone listening to a relaxing music (i.e., sounds from nature) with the purpose of both decreasing the stress and the initial excitement for the test and bringing all the subjects to a similar starting condition.

Cameras and physiological signal acquisition were started while the players were waiting. Any misplacing of sensors was checked by the operator from the external site by looking at the real time signals. The subjects have been instructed to minimize movements during the task to avoid artifacts. To increase subject involvement during the game, players have been told that they were competing for a prize. Prizes were given basing on a series of parameters including in-game performance, but also on physiological features, so that potential advantages of skilled player were reduced. Note that, from this moment on, to avoid the effect of covert communication [22], no further interaction between operator and subject occurred. The protocol was carried on by an automatic script on the computer that started each race and managed the questionnaire (see Section II-D).

After about one minute of relaxing music, the participants were asked to read the instructions and then, to start the trial by pressing a button. At the end of each race, starting from the second one, the participants were asked by a script to express, via a computer-based form, the preference between the race just played and the previous one. To minimize any potential order effect on physiological and self-reported data, each pair of game variants have been presented in both orders. The sequence of driver classes was as follows: W C L W L C W. With this sequence, all permutations pairs of classes W, C, L could be voted by the player once. The duration of each race was 3 minutes. This provided enough time to eliminate past race effects on physiological signals and to produce a new arousal level before the overcoming of boredom caused by excessive race length.

The total time of a session was about 30 minutes, i.e., 21 minutes (7 races  $\times$  3 min.) of racing and about 7 minutes of setup, question answering and resting.

### C. Acquired data

In this protocol four types of data have been acquired: physiological data, questionnaire answers (presented in Section II-D), game logs and video camera recordings.

Physiological data were gathered using the ProComp Infiniti device [23]. This device captured 5 physiological signals: BVP, Electrocardiogram (ECG), GSR, Respiration (RESP) and Temperature (TEMP). A sample rate of 256Hz

TABLE I  
EXTRACTED FEATURES

| Feature of $x$ | Description                               |
|----------------|---|
| $x_m$          | Mean                                      |
| $x_v$          | Variance                                  |
| $x_{min}$      | Min value                                 |
| $x_{max}$      | Max value                                 |
| $x_D$          | Max – Min                                 |
| $x_{tm}$       | Time of Min value                         |
| $x_{tM}$       | Time of Max value                         |
| $x_{dT}$       | $\Delta T$ of Max and Min                 |
| $x_{fd}$       | Mean Absolute value of first differences  |
| $x_{sd}$       | Mean Absolute value of second differences |
| $x_{ct}$       | Trend                                     |
| $x_{acf}$      | AutoCorrelation function at 10s           |

has been used except for ECG and BVP signals that were sampled at 2048Hz. The hand not used for interacting with the game was fitted with GSR, BVP and TEMP sensors. The 3 terminal ECG sensor were placed around the chest, as well as the RESP sensor.

Based on previous literature [24], [25], [26], [9], several derived signals have been extracted from the basic ones at the original sampling rate. Heart rate has been derived both from ECG ( $HR_{ecg}$ ) and BVP ( $HR_{bvp}$ ); magnitude (SM) and duration (SD) of signal variation has been derived from GSR; inspiration/expiration time ( $inTime$ ,  $outTime$ ), apnea in/out time ( $apneau$ ,  $apnealow$ ) and respiration interval ( $rTime$ ) have been extracted from respiration signal; upper/lower envelope of BVP ( $BVP_{up}$ ,  $BVP_l$ ) and their difference ( $BVP_d = BVP_{up} - BVP_l$ ) have been also computed.

A feature vector  $F = [f_1 f_2 \dots f_D] \in \mathcal{R}^D$  has been finally obtained by the union of features described in Table I computed for each mentioned signal during each race. We assume that the first part of each race is subject to transitory phenomena due to the transition from a race to the next one. Thus, these features have been computed by considering only the last 60 seconds of each race.

A log file containing timestamp and some game status variables was saved during each race. Note that information regarding the TORCS state has not been used to obtain the results reported in this paper, but the timestamps have been used for synchronization between races and physiological data. Two video cameras recorded the environment in which the player acted too. A frontal camera captured the player's face, the second camera was placed at the top right back corner of the room, with respect to player, and captured the player actions and the game output from the monitor. All captured frames have been associated with a timestamp that is used for synchronization with the other signals. These data have not been considered in the analysis presented in this paper, but will be used by further research activities.

### D. Questionnaire

Enjoyment preference between races have been collected. At the end of each race, the subject was asked whether he/she enjoyed more the last race or the previous one. A

pairwise preference scheme (2-alternative forced choice: 2-AFC) has been used in self reports. 2-AFC offers a main advantage to acquire objective enjoyment: it normalizes the different conception of enjoyment among subjects and it allows a fair comparison between the answers of different subjects. Since we are concerned with finding a general model for the relationship between physiological features and reported entertainment preferences that generalizes over different players, 2-AFC is preferred with respect to other approaches, such as ranking [27].

### III. METHODS

Because of the lack of absolute ranked answers about player subjective enjoyment, canonical classification methods based on learning a target output is inapplicable since the target output is not defined.

Several techniques that learn from a set of pairwise preferences exist. Such algorithms are based on Gaussian processes [28], support vector machines (SVNs) [29], and evolving artificial neural networks (ANNs) [9]. In this work we focused on a linear approach developing a linear classifier, based on subject reported preferences and physiological features, which has similar performance, simpler structure, and lower computational demand.

#### A. Preliminary Statistical Analysis

A statistical analysis has been performed to understand the relationship between physiological features and reported enjoyment. In the first part of this analysis a Pearson's chi-square test [30] has been performed to establish whether the null hypothesis of independence between a feature and the reported preference could be accepted or rejected. When the null hypothesis is rejected, the feature and the preference can be considered not independent, but the strength of the true relationship is still unknown.

Then, a correlation analysis has been performed as proposed in [9]. We use Cohen's Kappa coefficient [31] to evaluate correlation between features and reported enjoyment defined as follows:

$$k = \frac{Pr(a) - Pr(e)}{1 - Pr(e)} \quad (1)$$

where  $Pr(a)$  is the relative observed agreement between the user preference and the difference of the mean of a single physiological feature between a pair of races;  $Pr(e)$  is the hypothetical probability of chance agreement, using the observed data to calculate the probabilities of each feature to be randomly correlated. If the preferences and the variation of mean values between pair of races are in complete agreement then  $k = 1$  (or  $k = -1$  in case of anticorrelated features). If there is no correlation (other than what would be expected by chance) then  $|k| \leq 0$ .

#### B. Preference Learning

Preference learning [6] is a technique that aims to predict the subject's preference among different elements. A subject expresses a preference  $A \succ B$  (we say  $A$  is preferred

over  $B$ ) between two elements  $A$  and  $B$  when he/she is able to order them with respect to a personal preference criterion. Each element is characterized by a set of features  $F = [f_1 f_2 \dots f_D] \in \mathcal{R}^D$ . The goal of preference learning is to estimate a preference function  $P$  that respects the set of  $N$  constraints:

$$\text{if } A_i \succ B_i \quad i = 1 \dots N \quad \text{then } P(F_i^A) > P(F_i^B)$$

Where  $F_i^A$  and  $F_i^B$  are the features vectors of elements  $A$  and  $B$  respectively in the  $i$ -th comparison and  $N$  is the total number of comparisons. Preference learning is a general approach and can be used to model subject's preference from physiological data. There are different techniques that can be used to estimate  $P$  depending on the function used. The Large Margin Algorithm (LMA) [29] can be used as linear preference learning classifier, and comes from Support Vector Machine theory (SVMs). This algorithm considers a linear combination of individual features  $F$  as emotional preference function  $P(F) = FW^T$  where the weight vector  $W = [w_1 w_2 \dots w_D]$  binds the user preferences to the physiological features. This method was first applied by Fiechter and Rogers [29] to a routing problem where the particular structure of the problem lead to a simplification of SVM approach to a linear problem. The main simplification was given by the fact that, in routing problems, preference decreases with costs represented by features  $F$ . Thanks to this hypothesis it was possible to add a new set of constraints  $w_j \geq 0 \quad j = 1, \dots, D$  that brought the quadratic problem into a linear optimization problem.

This method has been applied also to model subject preferences from physiological features as reported in [9]. However, features in our problem cannot be considered as costs and thus, the hypothesis  $w_j \geq 0 \quad j = 1, \dots, D$  would lead to a non optimal solution: weights greater than zero are given only to positively correlated physiological features i.e., negatively correlated features are ignored. We propose then to use a linear approach for preference learning based on Linear Discriminant Analysis (LDA) A.K.A Fisher's projection [8].

Given a set of  $N$  race pairs  $R_i^A \succ R_i^B \quad i = 1, \dots, N$  where the subject prefers  $R_i^A$  over  $R_i^B$  the goal is to estimate  $W$  in such a way that the user preference is preserved:

$$\text{if } R_i^A \succ R_i^B \quad i = 1, \dots, N \quad \text{then } F_i^A W^T > F_i^B W^T$$

where  $F_i^A$  and  $F_i^B$  are the feature vectors associated to  $R_i^A$  and  $R_i^B$  respectively. We can rewrite the previous inequality as  $(F_i^A - F_i^B)W^T = F_i^d W^T > 0$ . Where  $F_i^d$  is the feature difference between preferred and not preferred races of pair  $i$ . We thus reformulate the problem of estimating  $W$  as a linear classification problem by considering the data set  $X = \{x_i | i = 1 \dots N\}$ ,  $C = \{c_i | i = 1 \dots N\}$ , where  $x_i = [F_i^d, -F_i^d]$  and  $c_i = 0, 1$  (i.e., we assign class 0 to positive examples  $F_i^d$  and class 1 to negative examples  $-F_i^d$ ). The problem can be solved with Fisher's projection, which finds a projection direction  $W$  in which classes are well separated. In this way,  $W$  can be used to predict one of the two classes by evaluating the inequality  $XW^T < K$  (in our case  $K=0$  since the mean of our data is 0).

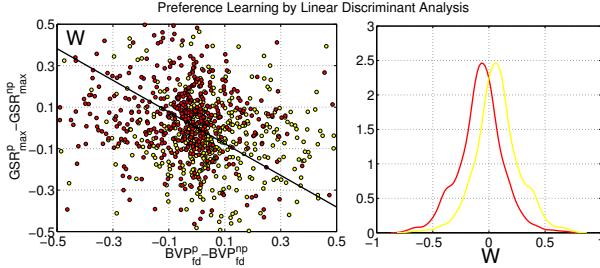


Fig. 1. Preference Learning by Linear Discriminant Analysis. A 2-D scatter plot of data from  $BVP$  and  $GSR$  is presented on the left. Sample points belonging to different classes have different colors. The black line  $W$  represents the direction computed by LDA in which data are projected. Probability densities of data by class with respect to the projection direction  $W$  is presented on the right. The direction  $W$  found by LDA is the one that best separates such densities.

To obtain Fisher's projection, we first define a within-class scatter matrix  $S_c$  as:

$$S_c = \sum_{c \in [0,1]} \sum_{x \in c} (x - \mu_c)(x - \mu_c)^T \quad (2)$$

where  $\mu_c$  is the mean value of sample points that belong to class  $c$ . Then we define a between-class scatter matrix

$$S_b = \sum_{c \in [0,1]} N_c (\mu - \mu_c)(\mu - \mu_c)^T \quad (3)$$

where  $N_c$  is the number of sample points in the class  $c$  and  $\mu$  is the mean value of all sample points. The best  $W$  that separates all the classes is the one that maximizes

$$J(W) = \frac{W^T S_c W}{W^T S_b W}. \quad (4)$$

Finding  $W$  is a one step process that requires much less time than running the optimization algorithm required by LMA.

Fisher's projection produces good classification performance when the distributions of data belonging to the same class are unimodal and classes are well separated. Figure 1 shows an example of application of LDA (based on real data obtained during our experiment) to classify the preferences of a subject by using two biological features  $BVP_{fd}$  and  $GSR_{max}$ . We can observe that both the distributions of  $BVP_{fd}^p - BVP_{fd}^{np}$  and  $GSR_{max}^p - GSR_{max}^{np}$  are unimodal but they are also partially overlapping, thus some of the preferences will be misclassified. Since all the data we are using in this work have an unimodal distribution similar to the one shown in Figure 1, the obtained performance depends on how well classes are separated.

#### IV. DATA ANALYSIS

In this section, the performance of the model for predicting reported preference is discussed. First we introduce the statistical analysis, then we present the results of the linear classifier based on LDA technique. The result will be finally extended by using different features selection methods.

#### A. Statistical Analysis

Results from the statistical analysis indicate that not all the features are dependent on the preference. This is confirmed by the fact that all p-values obtained from Pearson's  $\chi^2$  test are close to 0 ( $< 10^{-5}$ ); thus, we cannot accept the hypothesis of independence. The second part of the analysis characterizes the kind of dependence between features and preference by using the correlation coefficients  $k$ . The results of this analysis are presented in Table II where correlation coefficients  $k$  and  $\chi^2$  values from Pearson's  $\chi^2$  test are shown. The table is organized as follows: features derived from the same physiological signal are grouped together. For each group, the 5 most correlated features (higher absolute values of  $k$ ) are shown.

The physiological measurement that best correlates with reported user enjoyment is  $GSR$ , which achieved a correlation coefficient  $k = 0.332$  followed by  $BVP$  with  $k = -0.285$ . This result indicates that players tend to prefer games in which features from  $GSR$  increase and features from  $BVP$  decrease. Similar findings have been reported by Mandryk et al in [32] where  $GSR$  values resulted correlated with fun. Features related to respiration reported in Table II, represent time differences between respiration events and have also high negative correlation with enjoyment (i.e.,  $apnealow_m$  has a  $k = -0.234$ ). Thus, players preferred games in which breathing rate is increased (the time is decreased). Finally, good values of correlation have been obtained for  $HR$  ( $k = 0.166$ ) and temperature ( $k = -0.197$ ). Similar correlation results are reported also by Yannakakis and Hallam in [9] in their experiment on a physical playground. Our experiments have been performed on a computer video game that does not require physical activity, thus, features that best match user preferences in our work do not completely match the ones that were previously reported in [9].

#### B. Classification by Linear Discriminant Analysis

In the previous section, we have shown significant correlation between physiological features and the reported subject enjoyment. In this section, we present a quantitative evaluation of how each physiological feature can be used independently to predict the user preference. For each feature  $f_j$  ( $j = 1 \dots D$ ), an emotional preference function  $P(f_j) = f_j w_j$  is estimated through LDA technique [8] as explained in the previous section. Note that, since we are using only one feature, the estimated  $w_j$  could be only  $-1$  or  $1$  depending on the type of correlation. Given a pair of races  $R^A, R^B$ , the function  $P(f_j)$  classifies  $R^A$  as more entertaining if  $P(f_j^A) > P(f_j^B)$  where  $f_j^A$  and  $f_j^B$  are the  $j$ -th feature of preferred and non preferred race respectively. The performance of the preference function is defined as the number of correct pairwise classifications with respect to the total number of pairs (i.e., Correct Classification Rate CCR).

To guarantee significant values of performance, a leave-one-subject out cross validation has been applied to the classification process as follows: data relative to one subject

TABLE II

CORRELATION OF FEATURE AND PREFERENCE. HIGHER ABSOLUTE VALUES OF K MEAN HIGHER CORRELATION BETWEEN THE SINGLE FEATURE AND THE USER PREFERENCE.

|             | Bvp    |      |          | HR         |        |          | GSR         |       |          | Resp             |        |          | Temp        |        |          |
|-------------|--------|------|----------|------------|--------|----------|-------------|-------|----------|------------------|--------|----------|-------------|--------|----------|
|             | f      | k    | $\chi^2$ | f          | k      | $\chi^2$ | f           | k     | $\chi^2$ | f                | k      | $\chi^2$ | f           | k      | $\chi^2$ |
| $bvp_{fd}$  | -0.285 | 36.9 |          | $hr_{dT}$  | 0.166  | 12.5     | $gsr_{sd}$  | 0.332 | 49.7     | $apnealow_m$     | -0.234 | 24.6     | $temp_{ct}$ | -0.197 | 17.5     |
| $bvp_{sd}$  | -0.285 | 36.9 |          | $hr_m$     | 0.145  | 9.6      | $gsr_{fd}$  | 0.328 | 48.3     | $apnealow_{min}$ | -0.234 | 24.6     | $temp_{tm}$ | 0.188  | 15.9     |
| $bvp_v$     | -0.244 | 26.9 |          | $hr_{min}$ | 0.145  | 9.6      | $SM_{fd}$   | 0.310 | 43.2     | $rrate_{max}$    | -0.229 | 23.7     | $temp_v$    | -0.159 | 11.6     |
| $bvp_m$     | 0.232  | 25.2 |          | $hr_{max}$ | 0.111  | 5.6      | $SM_{sd}$   | 0.310 | 43.2     | $inrate_m$       | -0.213 | 20.4     | $temp_D$    | -0.161 | 11.8     |
| $bvp_{min}$ | 0.232  | 25.2 |          | $hr_{tm}$  | -0.103 | 4.7      | $gsr_{max}$ | 0.265 | 31.8     | $inrate_{min}$   | -0.213 | 20.4     | $temp_{fd}$ | -0.143 | 9.4      |

TABLE III

CORRECT CLASSIFICATION RATE USING LDA ALGORITHM AS LEARNING TECHNIQUE FOR SINGLE FEATURES.

|             | Bvp   |     | HR         |       | GSR         |       | Resp             |       | Temp        |       |
|-------------|-------|-----|------------|-------|-------------|-------|------------------|-------|-------------|-------|
|             | f     | CCR | f          | CCR   | f           | CCR   | f                | CCR   | f           | CCR   |
| $bvp_{fd}$  | 0.638 |     | $hr_{dT}$  | 0.582 | $gsr_{sd}$  | 0.667 | $apnealow_m$     | 0.616 | $temp_{ct}$ | 0.596 |
| $bvp_{sd}$  | 0.638 |     | $hr_m$     | 0.571 | $gsr_{fd}$  | 0.664 | $apnealow_{min}$ | 0.616 | $temp_v$    | 0.582 |
| $bvp_v$     | 0.618 |     | $hr_{min}$ | 0.571 | $SM_{fd}$   | 0.656 | $rrate_{max}$    | 0.611 | $temp_D$    | 0.582 |
| $bvp_m$     | 0.613 |     | $hr_{max}$ | 0.556 | $SM_{sd}$   | 0.656 | $inrate_m$       | 0.604 | $temp_{fd}$ | 0.573 |
| $bvp_{min}$ | 0.613 |     | $hr_{tm}$  | 0.551 | $gsr_{max}$ | 0.636 | $inrate_{min}$   | 0.604 | $temp_{sd}$ | 0.569 |

have were and remaining data are used for training; the LDA is trained on the training data set and the performance of the model is tested on the data of the removed subject. These steps are iterated for each subject and the mean values are reported in Table III. The table is organized as follows: features derived from the same physiological signal are grouped together. For each group are shown the 5 features that give the best classification performance using LDA. Note that the values are consistent with the analysis of correlation reported in Table II: The best classification performance ( $CCR = 0.667$ ) is achieved by GSR, which is the most correlated signal having the highest absolute value of  $k$ . BVP obtained a  $CCR=0.638$ , respiration reported  $CCR=0.616$  and finally, for HR and temperature the  $CCR$  was 0.582 and 0.596, which are closer to 0.5 characteristic of a random classifier.

The results confirm that the Preference Learning approach can be successfully applied to model player preference in video games where the physical activity is kept as constant as possible over different type of games. Moreover, these results support the proposed experimental protocol as a successful way to produce reliable and replicable data for affective computing experiments.

### C. Improvements by Considering Multiple Features

To improve single feature performance, a linear combination of all physiological features  $F$  by LDA has been used to predict subject enjoyment as explained in previous section. This is a generalization of the analysis performed over a single feature since a combination of features may introduce information useful for classification. Leave-one-subject out cross validation has been used to evaluate the performance. The LDA algorithm takes the full set of features as input and tries to find the best combination of weights  $W$  that separates classes. Correct classification rate achieved using all features

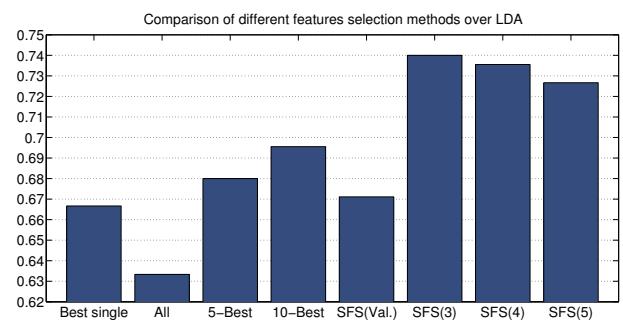


Fig. 2. Classification performance achieved with different subset of features: Best single feature, All features, Selection of best 5 or 10 features (5-Best and 10-Best), SFS on validation data (SFS(Val.)), SFS with pseudo-intersection ( $n=3,4,5$ )

is 0.633. This result is lower than the one obtained with the best single feature. This is likely, since some of the features might have introduced noise instead of adding useful information.

Since including all features did not provide better performance than the best single feature, we tried to find a subset of features that performed better than single feature classification. We selected the first 5 and 10 features that individually gave the best CCR and the performance achieved was 0.68 for 5-best and 0.696 for 10-best. CCR is higher than the one obtained with the complete set likely because some of the noisy features are not included by this rough selection technique. A comparison of performances is reported in Figure 2

A better, still greedy, method for feature selection is the Sequential Feature Selection (SFS) [33] that finds the minimal subset of features that maximize the classification performance. It is a bottom-up algorithm where, at each

step, an additional feature is added to the current feature set. The selected feature is added if its marginal value with respect to the classification function is positive. To evaluate the performance of a feature selection algorithm like SFS, we used an external cross-validation procedure. This avoids the overfitting of selection for the current dataset. A K-fold (K=15) cross-validation by subject has been performed. Data belonging to all the players in the original dataset  $D_s$  have been randomly split into K folds  $D_{s_i}$   $i = 1 \dots 15$  (containing data of 5 subjects each). Folds have been then assembled into 15 data set containing training data  $D_{s_i}^{tr} = D_s \setminus D_{s_i}$  and validation data  $D_{s_i}^{val} = D_{s_i}$ . The SFS has been executed K times independently over each different data set. For each iteration, the best feature selection with respect to the training data  $D_{s_i}^{tr}$  has been evaluated on validation data  $D_{s_i}^{ts}$  in order to estimate the selection performance on previously unseen data. Each selection found on training has been evaluated with the same leave-one-subject out cross validation method presented in Section IV-A. Each independent run selected the best subset of features that obtained different performance on validation data with mean value 0.671 of correct preference prediction over all the data set. Due to the different type of cross validation, this estimate of performance results the most general and least overfitted, hence the most significant. This result, labeled as SFS(Val.), is compared in Figure 2 with other feature selection methods.

The performance over validation gives an estimation of how well the classifier would perform with previously unseen data. However, with the presented SFS approach it is not possible to know which is the best subset of features that maximizes the CCR since each fold may yield to a different selection. To give an indicative measure of performance of the best general subset of features, a pseudo-intersection of the selections produced by each of the 15 independent runs has been performed. The pseudo-intersection merges the features that have been selected by at least  $n$  independent runs of SFS. We compared pseudo-intersection feature subsets with  $n = 3, 4, 5, 6, 7$ . We finally tested the obtained selections over the full data set with the previous described leave-one-subject out cross validation. In Figure 2 SFS(3)=0.74, SFS(4)=0.7356 and SFS(5)=0.7267 represent results obtained by the pseudo-intersections when features where selected by 3, 4 or 5 runs out of 15. The highest value is obtained with SFS(3) since it uses almost all the features that each fold have selected. This is the highest performance we achieved through leave-one-subject out cross validation by applying a linear function for preference modeling on these data.

In Table IV, we reported the features selected by some of the used methods. In the case of pseudo-intersection the performance decreases as much as we increase the number of folds from which features have been selected. This is due to the fact that pseudo-intersection exploits the information coming from each independent run. The less runs are considered, the more performance is overestimated, since we are using more data to obtain the selection. However, the more a feature has been selected from a fold the more

TABLE IV  
SELECTED FEATURES BY DIFFERENT METHODS

| Type   | Selection   | Performance |
|--------|---|-------------|
| SFS(3) | $BVP_{fd}$ $BVP_{sd}$ $GSR_{fd}$ $GSR_{sd}$ $SM_v$<br>$SD_{fd}$ $inrate_m$ $outrate_v$ $apnealow_v$ | 74%         |
| SFS(4) | $BVP_{fd}$ $GSR_{fd}$ $SM_v$ $SD_{fd}$ $outTime_v$  | 73.56%      |
| SFS(5) | $BVP_{fd}$ $GSR_{fd}$ $SD_{fd}$ $outTime_v$   | 72.67%      |
| SFS(6) | $BVP_{fd}$ $GSR_{fd}$ $SD_{fd}$ $outTime_v$   | 72.67%      |
| SFS(7) | $BVP_{fd}$ $GSR_{fd}$ $outTime_v$   | 69.33%      |
| SFS(8) | $GSR_{fd}$  | 66.67%      |
| 5-Best | $BVP_{fd}$ $GSR_{sd}$ $GSR_{fd}$ $SM_{sd}$ $SM_{fd}$  | 68%         |

it is an invariant measure of preference among subjects. This is the case of  $BVP_{fd}$ ,  $GSR_{fd}$  and  $outTime_v$  that have been selected at least by 7 out of 15 runs of SFS, so they are likely to be invariant features among subjects to predict enjoyment. Table IV shows also how the selections produced by SFS are different with respect to the 5-Best or 10-Best approach. SFS obtains higher performance, fixed the number of used features (i.e., SFS(4) vs 5-Best), since it removes variables that are dependent on the ones that have been already selected (i.e.,  $GSR_{sd}$  and  $GSR_{fd}$  or  $SM_{sd}$  and  $SM_{fd}$ ) and introduces other variables that can be more useful even if they do not produce high performance when used alone.

By means of this analysis we have now a clear picture of which are the most relevant features that combined linearly can predict the reported preference of video game players with a CCR up to 0.74% on previously ~~unseen~~ data.

## V. CONCLUSION

We have presented a way to identify the preference of players among different variants of a video game. We have proposed a new video game scenario in which 3 different game conditions have been obtained by controlling the opponent skill. The proposed scenario requires the player to be sitting in front of a computer, therefore the effects of movement artifacts on acquired data are reduced. Moreover, thanks to the adaptive controller, the game experience during the test has been homogeneous among different subjects. A data set composed by physiological data, questionnaire answers regarding user data, game experiences, race preferences, game logs, 2 video camera recordings have been acquired with the purpose of building a benchmark data set in which different Affective Computing techniques can be applied and validated.

We performed correlation analysis between physiological data and subject preference, showing that features from  $GSR$  have a high correlation with player reported preferences ( $k = 0.332$ ,  $\chi^2 = 49.7$ ,  $p-value \approx 10^{-5}$ ). Similar results have been shown by Mandryk et al. in [32] and are slightly different from the ones presented by Yannakakis and Hallam in [9] probably due to the different nature of the task involved (computer video game vs physical playground). We have estimated a linear model of preference that maps a subset of physiological features to the preference level, through a novel approach that makes use of Linear Discriminant Analysis (LDA). Results showed that this model is able to

predict reported preference with an accuracy up to 0.74% on previously unseen data.

The analysis of questionnaires highlighted that for less than 42% of subjects there was a consistent agreement on the order of preference of game variants. That means that for them, a situation where the opponent is as skilled as the player was always preferred. However, the remaining players either did not answered coherently among different repetition of the same stimulus or they expressed a different order of personal preference. This interesting result indicates that the game preference is personal and it is hard to design a priori game experience (e.g., by changing opponent skills) that results suitable for everyone. This motivates the current research that aims to evaluate the enjoyment from biological signals without any assumption on players game preferences.

Player satisfaction from physiological data, is perhaps one of the most promising application area of affective computing. Classic and canonical games could be enhanced to adapt to the player affective states, and entirely new types of games could be created. Results from this experiments can be used as starting point for a follow up experiment in which game experience is modified in realtime with the purpose of keeping the player satisfaction to a high level.

#### ACKNOWLEDGMENTS

The research activity described in this paper has been partially supported by IIT - Italian Institute of Technology.

#### REFERENCES

- [1] P. Ekman, R. Levenson, and W. Friesen, "Autonomic nervous system activity distinguishes among emotions," *Science*, vol. 221, no. 4616, pp. 1208–1210, 1983.
- [2] "The open racing car simultaor website," <http://torcs.sourceforge.net/>.
- [3] G. Yannakakis and J. Hallam, "Towards capturing and enhancing entertainment in computer games," *Lecture Notes in Computer Science*, vol. 3955, p. 432, 2006.
- [4] G. Yannakakis, H. Lund, and J. Hallam, "Modeling children's entertainment in the playware playground," in *Proceedings of the IEEE Symposium on Computational Intelligence and Games*, 2006, pp. 134–141.
- [5] J. Furnkranz and E. Hullermeier, "Pairwise preference learning and ranking," *Lecture Notes in Computer Science*, vol. 2837, pp. 145–156, 2003.
- [6] J. Doyle, "Prospects for preferences," *Computational Intelligence*, vol. 20, no. 2, pp. 111–136, 2004.
- [7] G. Yannakakis, "Preference Learning for Affective Modeling," in *Proceeding of the international conference on Affective Computing and Intelligent Interaction, ACII 2009*, Amsterdam, Netherland, 2009.
- [8] R. Duda, P. Hart, et al., *Pattern classification and scene analysis*. Wiley New York, 1973.
- [9] G. Yannakakis and J. Hallam, "Entertainment modeling through physiology in physical play," *International Journal of Human-Computer Studies*, vol. 66, no. 10, pp. 741–755, 2008.
- [10] ——, "Capturing Player Enjoyment in Computer Games," *Computational Intelligence (SCI)*, vol. 71, pp. 175–201, 2007.
- [11] T. Malone, "What makes computer games fun?" in *Proceedings of the joint conference on Easier and more productive use of computer systems.(Part-II): Human interface and the user interface-Volume 1981.* ACM New York, NY, USA, 1981.
- [12] M. Csikszentmihalyi, *Flow: The psychology of optimal experience*. Harper & Row New York, 1990.
- [13] J. Cacioppo, L. Tassinary, and G. Berntson, *Handbook of Psychophysiology*. New York, NY: Cambridge University Press, 2000.
- [14] R. Picard, E. Vyzas, and J. Healey, "Toward machine emotional intelligence: analysis of affectivephysiological state," *IEEE transactions on pattern analysis and machine intelligence*, vol. 23, no. 10, pp. 1175–1191, 2001.
- [15] C. Lisetti, F. Nasoz, C. LeRouge, O. Ozyer, and K. Alvarez, "Developing multimodal intelligent affective interfaces for tele-home health care," *International Journal of Human-Computer Studies*, vol. 59, no. 1-2, pp. 245–255, 2003.
- [16] J. Kim and E. Andrè, "Emotion recognition based on physiological changes in listening music," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 30 (12), pp. 2067–2083, December, vol. 30, no. 12, pp. 2067–2083, December 2008.
- [17] R. Mandryk, K. Inkpen, and T. Valvert, "Using psychophysiological techniques to measure user experience with entertainment technologies," *Behaviour & information technology(Print)*, vol. 25, no. 2, pp. 141–158, 2006.
- [18] R. Mandryk and M. Atkins, "A fuzzy physiological approach for continuously modeling emotion during interaction with play technologies," *International Journal of Human-Computer Studies*, vol. 65, no. 4, pp. 329–347, 2007.
- [19] P. Rani, N. Sarkar, and C. Liu, "Maintaining optimal challenge in computer games through real-time physiological feedback," in *Proceedings of the 11th International Conference on Human Computer Interaction*, 2005, pp. 184–192.
- [20] T. Tijs, D. Brokken, and W. IJsselsteijn, "Dynamic game balancing by recognizing affect," in *Proceedings of the 2nd International Conference on Fun and Games*. Springer, Berlin, 2008, p. 93.
- [21] S. McQuiggan, S. Lee, and J. Lester, "Predicting user physiological response for interactive environments: an inductive approach," in *Proceedings of the 2nd Artificial Intelligence for Interactive Digital Entertainment Conference*, 2006, pp. 60–65.
- [22] R. Rosenthal, "Covert communication in laboratories, classrooms, and the truly real world," *Current Directions in Psychological Science*, pp. 151–154, 2003.
- [23] "Thought technology ltd., 2002. website," <http://www.thoughttechnology.com/>.
- [24] A. Bonarini, L. Mainardi, M. Matteucci, S. Tognetti, and R. Colombo, "Stress recognition in a robotic rehabilitation task," in *Proc. of "Robotic Helpers: User Interaction, Interfaces and Companions in Assistive and Therapy Robotics"*, a Workshop at ACM/IEEE HRI 2008, vol. 1. Amsterdam, the Netherlands: University of Hertfordshire, March 2008, pp. 41–48.
- [25] S. Tognetti, C. Alessandro, A. Bonarini, and M. Matteucci, "Fundamental issues on the recognition of autonomic patterns produced by visual stimuli," in *Proceeding of the international conference on Affective Computing and Intelligent Interaction, ACII 2009*, Amsterdam, Netherland, 2009.
- [26] R. Picard, E. Vyzas, and J. Healey, "Toward machine emotional intelligence: Analysis of affective physiological state," *IEEE transactions on pattern analysis and machine intelligence*, pp. 1175–1191, 2001.
- [27] R. Likert, "A technique for the measurement of attitudes," *Archives of Psychology*, vol. 140, pp. 1–55, 1932.
- [28] W. Chu and Z. Ghahramani, "Preference learning with Gaussian processes," in *Proceedings of the 22nd international conference on Machine learning*. ACM, 2005, p. 144.
- [29] C. Fiechter and S. Rogers, "Learning subjective functions with large margins," in *ICML 2000: Proceedings of the Seventeenth International Conference on Machine Learning*, 2000, pp. 287–294.
- [30] H. Chernoff and E. Lehmann, "The use of maximum likelihood estimates in  $\chi^2$  tests for goodness of fit," *The Annals of Mathematical Statistics*, pp. 579–586, 1954.
- [31] J. Cohen, "Coefficient of agreement for nominal scales. Educational and Psychological Measurement," *Psychological bulletin*, vol. 20, pp. 37–46, 1960.
- [32] R. Mandryk, M. Atkins, and K. Inkpen, "A continuous and objective evaluation of emotional experience with interactive play environments," in *Proceedings of the SIGCHI conference on Human Factors in computing systems*. ACM, 2006, pp. 1027–1036.
- [33] P. Pudil, J. Novoviová, and J. Kittler, "Floating search methods in feature selection," *Pattern recognition letters*, vol. 15, no. 11, pp. 1119–1125, 1994.

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/225407995>

# Ranking vs. Preference: A Comparative Study of Self-reporting

Conference Paper · October 2011

DOI: 10.1007/978-3-642-24600-5\_47 · Source: dx.doi.org

---

CITATIONS

65

READS

109

2 authors, including:



Georgios Yannakakis

University of Malta

220 PUBLICATIONS 5,360 CITATIONS

[SEE PROFILE](#)

Some of the authors of this publication are also working on these related projects:



AutoGameDesign -- Sonancia Project [View project](#)



COMnPLAY-Science [View project](#)

# Rating vs. Preference: A comparative study of self-reporting

Georgios N. Yannakakis<sup>1</sup> and John Hallam<sup>2</sup>

<sup>1</sup> Center for Computer Games Research, IT University of Copenhagen, Rued Langgaards Vej 7, Copenhagen S, Denmark [yannakakis@itu.dk](mailto:yannakakis@itu.dk)

<sup>2</sup> Maersk Mc-Kinney Moller Institute, University of Southern Denmark, Campusvej 55, Odense, Denmark [john@mmt.sdu.dk](mailto:john@mmt.sdu.dk)

**Abstract.** This paper introduces a comparative analysis between rating and pairwise self-reporting via questionnaires in user survey experiments. Two dissimilar game user survey experiments are employed in which the two questionnaire schemes are tested and compared for reliable affect annotation. The statistical analysis followed to test our hypotheses shows that even though the two self-reporting schemes are consistent there are significant *order of reporting* effects when subjects report via a rating questionnaire. The paper concludes with a discussion of the appropriateness of each self-reporting scheme under conditions drawn from the experimental results obtained.

## 1 Introduction

Self-reporting provides the most direct approach to user experience annotation and affect detection. Quantitative reports via questionnaires offer unique properties for constructing computational models of reported user states (affective or cognitive) and ease the analysis of subjective assessment in user studies. Even though beneficial for cognitive and affective capture and modeling, such reporting has several limitations such as self-deception, intrusiveness and subjectiveness. The appropriateness of the reporting scheme used for affect detection is therefore vital for the validity of the obtained analysis.

This paper examines the relationship between two popular self-reporting schemes in user studies: self-reporting via *rating* (or scaling) and via *pairwise preference*. The two schemes are compared in two dissimilar game survey studies in which experiment participants are asked to post-report a set of affective states. For the comparison to be possible, pairwise preferences are inferred from the rating values and compared to the direct pairwise preferences. The two hypotheses the two questionnaire schemes are tested against are:

- H1: There is an inconsistency between reported preferences and reported rating.  
Rating responses do not match reported preferences.
- H2: The order of post-experience reporting has an effect on both rating and preference report schemes. Randomness exists in both self-report schemes.

The statistical analysis followed to test the above hypotheses suggests that while rating and preferences are consistent (with variant degrees of consistency), pairwise preferences are more appropriate detectors of user states, eliminating the subjective notion of scaling and effects related to reporting order.

## 2 Self-reporting

This paper focuses on **forced self-reports** obtained via questionnaires. Such a self-report scheme constrains the participant to specific questionnaire items which could vary from simple tick boxes to multiple choice items while both the questions and the answers provided could vary from single words to sentences. Two types of forced self-reports that are described in more detail below, define the framework of investigations in this paper: self-reports via **rating** (or scaling) and self-reports via **preferences**.

### 2.1 Rating

The vast majority of psychometric and user studies have adopted a type of rating report to capture the subjective assessment of the experiment participants ([11] among others). The most popular approach to rating reports is a form of a **Likert scale** [5] in which users are asked to rate an experience, an emotion or an interactive session. In most such studies Likert ratings are **usually averaged across users** before they are further analyzed. Such a practice has an impact on the ratings **losing their subjective nature** but also implies a knowledge of the scale that is beyond a relative rank order of data [11].

Among the limitations of rating ordinal scales, Linn and Gronlund [6] indicate the existence of personal bias which may (among others) occur when the subject is consistently using only part of the scoring scale, logical errors due to the confusion of the distinct items of an ordinal scale and the ability to use numerical information within scales which is affected by the subject's internal cognitive processes, cultural background, temperament, and interests [13]. There is also a large body of work suggesting the presence of *primacy* and *recency* order effects in Likert questionnaires (see [2] among others).

The authors are not aware of a reliable statistical test that validates the reliability of a rating questionnaire as a whole. Cronbach's alpha [4] (*inter alia*) is an estimate of internal consistency (or reliability) of sections of the questionnaire; Cohen's kappa [3] assesses rater agreement in nominal scales.

### 2.2 Preference

Reporting via pairwise preferences has recently attracted the interest of researchers in affective and cognitive modeling ([16, 14, 12] among others) since it minimizes the assumptions made about subjects' notions of highly subjective constructs such as emotions and allows a fair comparison between the answers of different subjects. Moreover, artifacts such as the subjective notion of rating/scaling are eliminated and lead to the construction of **generalisable and accurate computational models of affect via user preference modeling** [14].

A preference questionnaire scheme may ask for the pairwise or multiple preference of participants or even ask them to provide a preferred order. In this paper we investigate pairwise preferences and are inspired by the seminal work of Scheffe [10] and Agresti [11] for the analysis of paired comparisons.

### 3 User survey case studies

This section presents the main phases of the experimental procedure followed to obtain self-reported emotional or cognitive states of experiment participants via both rating and preference schemes. The reader is referred to [16] for more details on the experimental protocol used. The section concludes with the presentation of the two case studies considered in this paper.

#### 3.1 System Instrumentation

The interactive systems we investigated are instrumented based on controllable parameters identified by the designer. The selection of the parameters is based on their potential impact on the user's affective and cognitive states examined and thereby to the post-experience self-reporting. For instance, a controllable parameter in a game system could be the speed of the game.

For each parameter under investigation, a number of states (e.g. 'Low', 'High') are selected. The product of the number of states for each of the parameters defines the number of different system variants that will be examined. Given the proposed experimental design [16] each survey participant interacts with system variants in pairs (variant A and variant B) — differing in the levels/states of one or more of the selected controllable parameters — for a selected time window. To test for potential order effects each subject interacts with the aforementioned system variants in both orders. Each time a system variant is completed the subject is asked to rate a particular experience using both a rating and a pairwise preference reporting scheme (described below).

#### 3.2 Self-reported Post-experience

For rating questionnaires the question is expressed as: "The session felt  $E$ ." where  $E$  is the emotional state (e.g. frustration) under investigation. Two rating scales have been used in the experiments reported: a 20 point 0-10 scale, and a 1-5 scale. The 0-10 scale uses principles of the *funometer* [9]; subjects have to rate the experience in a thermometer-designed Likert scale. On the other hand, the answers in the 1-5 scale rating scheme are inspired by the game experience questionnaire (GEQ) [8]; numbers have following glosses: 1: *not at all*; 2: *slightly*, 3: *moderately*, 4: *fairly* and 5: *extremely*.

For pairwise preference questionnaires subjects are asked to fill in a questionnaire each time a pair of game sessions (variants) is finished. According to this scheme, the subject is asked to report whether the first variant felt more  $E$  than the second variant. Specifically, for each completed pair of system variants  $A$  and  $B$ , subjects report their preference regarding an emotional state,  $E$ , by selecting among the following 4-alternative forced choices (4-AFC):  $A$  [ $B$ ] felt more  $E$  than  $B$  [ $A$ ] (cf. 2-alternative forced choice); both felt equally  $E$ ; neither of the two felt  $E$ .

One of the limitations of the experimental protocol proposed is **post-experience**. Users report emotional states *after* playing games, which might generate memory-dependencies in the reports. Effects such as order of play and game learnability might also be apparent and interconnected to memory. The experimental protocol, however, is designed to test for order of play effects which, in part, reveal memory (report consistency over different orders) and learnability effects, if any. Lack of significant order effect provides evidence that the experimental noise generated in this way is random. Statistical analysis of the effect of order on subjects' emotional judgement indicates the level of randomness in subjects' preferences. **Randomness** is apparent when the subject's expressed preferences are inconsistent for the pair ( $A, B$ ) independently of the questionnaire-scheme used.

### 3.3 The Playware Case Study

The first case study presented concerns game play sessions followed by self-reporting sessions of children playing physical interactive games [16, 17]. The game, called '**Bug-Smasher**', designed using the Playware playground (interactive tiles) platform [7], is used here as the test-bed interactive system for investigating the relationship between self-reporting schemes. (The reader is referred to [16] for more details of Bug-Smasher).

Seventy six children, aged 8 to 10 years old, participated in the survey experiment. Each subject played a set of 90 second Bug-Smasher variants, differing with respect to two control parameters: the speed of the game and the spatial diversity of game opponents. Children were not interviewed but were asked to fill in a questionnaire, minimizing interviewing effects. Each subject was asked to rate each game via a 10-scale *funometer* [9] (in increments of 0.5) and after a pair of games were finished, to report a fun preference for the two games she played using a 2-AFC question, "which one of the two games was more fun?" The options offered for choice were "first" and "second".

### 3.4 The Maze-ball Case Study

A screen-based computer game, named Maze-ball, is used for the second experiment reported in this paper. **Maze-ball** [18] is a three-dimensional predator/prey game. The goal of the player (ball) is to maximize her score by gathering as many tokens, scattered in the maze, as possible while avoiding being touched by a number of opponents in a predefined time window of 90 seconds. Further details about Maze-Ball and experimental design can be found in [18].

Thirty six subjects aged from 21 to 47 years participated to the experiment. Each subject played a predefined set of eight games for 90 seconds each; the games differ in the virtual camera profile embedded. For each completed game and pair of games  $A$  and  $B$ , subjects report their emotional preference using a 5-point Likert scale based on GEQ [8] followed by a 4-AFC pairwise preference protocol. The emotional states,  $E$ , examined comprise *fun*, *challenge*, *boredom*, *frustration*, *excitement*, *anxiety* and *relaxation*. The selection of these seven states is based on their relevance to computer game-playing and player experience.

### 3.5 Case Study Dissimilarities

The main dissimilarities between the two case studies are that in Playware 1) subjects are children (aged: 8 to 10), 2) a pen-and-paper (instead of a digital) questionnaire is used, 3) a rather broad ordinal scale from 1 to 10 is used for the rating scheme, 4) 2-AFC (instead of 4-AFC) is used for the preference scheme; 5) and subjects are asked only one question, about fun. Cognitive load during the reporting phase in the Playware experiment appears less due to the presence of only one question. Moreover, the broad rating scale used may allow for a better approximation of the level of reported fun.

Comparison of findings across the two case studies is not appropriate given the large number of dissimilarities in terms of gameplay interaction and experimental protocol. However, collectively, they provide two related but different studies of post-experience reporting in games and their analysis assists the understanding of the interplay between reported preferences and rating across different schemes.

## 4 Results and analysis

This section presents the results of the statistical analysis for testing our hypotheses in the two case studies. First, the statistics employed to test our research hypotheses H1 and H2 are outlined below.

### 4.1 H1 test statistic

To measure the degree of agreement between the rating and preference self-reports we calculate the correlation coefficients between them, obtained using  $c(\mathbf{z}) = \sum_{i=1}^N \{z_i/N\}$  following the statistical analysis procedure for pairwise preference data introduced in [15].  $N$  is the total number of incidents to correlate, and  $z_i = +1$ , if rating reports match preference reports and  $z_i = -1$ , if rating and preference reports are mismatched in the game pair  $i$ . In the calculation of  $c(\mathbf{z})$  we only take into account *clear* preferences and ratings of participants. That is, we only consider game pairs in which both a clear preference (i.e.  $A > B$  or  $A < B$ ; 2-AFC) and a clear rating (i.e.  $A > B$  or  $A < B$ ) are expressed. The p-values of  $c(\mathbf{z})$  are obtained via the binomial distribution.

### 4.2 H2 test statistics

To measure whether the order of play affects the player's judgement of rating or pairwise preference for affective states, we follow the order testing procedure described in [15], based on the number of times that the subject prefers the first (primacy effect) or the second (recency effect) game in both pairs. Briefly, the order test statistic is calculated as  $r_o = (K - J)/N$ , where the subject prefers (either via rating or preference) the first session in both pairs  $K$  times and, the second session in both pairs  $J$  times. The greater the absolute value of  $r_o$  the more the order of play tends to affect the subjects' judgement of interest.  $r_o$  is trinomially-distributed under the null hypothesis..

In addition to the  $r_o$  value we calculate the  $r_c = (K + J)/N$  test statistic, which yields a measure of reporting consistency with respect to order. The obtained  $r_c$  value

lies between 0 (reporting is consistent) and 1 (reporting is inconsistent) and is binomially-distributed with mean 0.5 under the null hypothesis.

The order effects are calculated solely on clear preferences (i.e. when  $A \succ B$  or  $A \prec B$ ) and ratings (i.e. when  $A > B$  or  $A < B$ ) in both pairs played in both orders. The significance level used in this paper is 5%.

### 4.3 Playware

The total number of game pairs with valid reported data is 105 in the Playware experiment. To calculate the statistics we exclude the 35 game pairs in which an equal rating is reported. The correlation between reported rating and preference  $c(z) = 0.857$  ( $p$ -value  $= 4.002 \cdot 10^{-10}$ ) indicates a statistically significant effect and rules out H1.

**Order Effect Analysis** Statistical analysis of the subjects' answers shows that no significant order effect occurs ( $r_o = -0.102$ ,  $p$ -value = 0.224) when preferences are reported, which rules out hypothesis H2. However, a significant effect of playing order on rating reports is found ( $r_o = -0.3809$ ,  $p$ -value = 0.0097) which indicates a tendency to consistently rate the second game higher. The insignificant order effect for reported *preferences*, in part, demonstrates that effects such as a subject's possible preference for the very first game played and the interplay between reported fun and familiarity with the game are statistically insignificant. On the other hand, the significant order effect for reported rating suggests that the order of play influences reporting when the rating scheme is used.

The  $r_c$  values for rating and preferences are 0.476 ( $p$ -value = 0.124) and 0.338 ( $p$ -value = 0.009), respectively, suggesting that only the preference reports appear to be ~~consistent with respect to order~~.

**Analysis & Conclusions** The first case study provides indications of **inconsistency between rating and preference reports**. While the two are statistically correlated ( $c(z) = 0.857$ ) there are several instances (16.6% of the data samples) in which preferences do not agree with their corresponding rating.

The inconsistency between the two report schemes may have occurred for a number of reasons including self-deception, cognitive load, question understanding in small children etc. A first analysis of the effect of order of game interaction shows that significant order effects exist only in reported ratings which in turn suggests existence of **randomness** when expressing rating choices for the game sessions attempted. Moreover, the consistency of reports with respect to order,  $r_c$ , appears to be significant for the preference reports only.

### 4.4 ~~Maze-Ball~~

For the Maze-Ball case study we follow the same statistical analysis presented above for the Playware game. The total number of valid game pairs examined in the Maze-Ball survey is 56 and the matching correlation ( $c(z)$ ) values between rating and preference reports for the Maze-Ball test-bed are depicted in Table 1.

**Table 1.** Maze-ball: Correlation coefficient values ( $c(z)$ ) between rating and clear preferences (2-AFC), and order of play ( $r_o$ ) and consistency ( $r_c$ ) correlation coefficients for all investigated emotional states  $E$ . Significant effects appear in bold.

| $E$         | $c(z)$       | $r_o$         |            | $r_c$        |              |
|-------------|--------------|---------------|------------|--------------|--------------|
|             |              | Rating        | Preference | Rating       | Preference   |
| Fun         | <b>0.925</b> | <b>-0.375</b> | -0.150     | 0.375        | 0.450        |
| Challenge   | <b>0.733</b> | <b>0.300</b>  | -0.222     | 0.500        | 0.444        |
| Frustration | <b>0.878</b> | -0.083        | -0.066     | <b>0.250</b> | <b>0.187</b> |
| Anxiety     | <b>0.619</b> | 0.200         | -0.222     | <b>0.200</b> | 0.444        |
| Boredom     | <b>0.666</b> | <b>-0.333</b> | -0.111     | 0.333        | <b>0.111</b> |
| Excitement  | <b>0.642</b> | -0.200        | -0.117     | <b>0.200</b> | <b>0.312</b> |
| Relaxation  | <b>0.652</b> | <b>-0.250</b> | 0.052      | 0.250        | 0.368        |
| Total       | <b>0.744</b> | -0.090        | -0.112     | <b>0.309</b> | <b>0.353</b> |

It appears there is a varying degree of consistency between rating and preference reports depending on the affective state (question asked). Overall 2-AFC preference reports appear to be consistent with rating reports. For the fun, frustration and challenge reports the two schemes are highly correlated (correlation higher than 0.7) whereas for the other four questionnaire items the correlation lies within the 0.6-0.7 interval; however, in all seven affective state questionnaire items, the correlation is statistically significant ruling out H1. These effects might be linked to the order of question items appearing in the questionnaire which is equivalent to the order the emotional states that appear in Table 1; the questions about excitement and relaxation, for instance, were the last two items in both questionnaires.

**Order Effect Analysis** The statistical analysis presented in Table 1 shows that order of play does not affect the pairwise preferences of users. The insignificant order effects also, in part, demonstrate that effects such as a user's possible preference for the very first game played and the interplay between reported emotions and familiarity with the game are statistically insignificant. Even though not statistically significant, the correlation statistic values of Table 1 reveal a preference for the second game played for most questionnaire items (negative correlation values).

The H2 hypothesis is ruled-out: no effect exists in any preference questionnaire item while significant effects are observed in the fun, challenge, boredom and relaxation rating questions. These effects may, in part, explain the low  $c(z)$  values in boredom and relaxation but also be responsible for the level of inconsistency in fun and challenge. In general it appears that — excluding the anxiety state —  $c(z)$  values (significant or not) are larger in the rating scheme than in the preference questionnaire scheme. The total order effect is not significant for either questionnaire scheme, which does not allow any safe conclusions to be drawn when all questionnaire items are considered.

The  $r_c$  values in Table 1 demonstrate that both questionnaire schemes are consistent in frustration and excitement and no additional conclusions can be drawn for these two states. On the other hand, it appears as if the inconsistencies of anxiety preferences have

an impact on the low  $c(z)$  value of that state given that the  $r_c$  values are not significant. The order statistics computed including the equal preference (3-AFC) could provide a clearer picture of the relationship between order effects and questionnaire scheme inconsistencies and are left for future analysis due to space considerations.

**Analysis & Conclusions** The statistical analysis for the Maze-Ball case study revealed two main effects: consistency (of varying degree) between rating and preference reports in all 2-AFC questionnaire items and significant order effects for the rating scheme.

Results related to the first effect suggest that even though for some question items (e.g. fun, frustration and challenge) the consistency is higher than others (e.g. anxiety, relaxation and boredom), the hypothesis H1 is ruled out for all emotional states in Maze-Ball. Nevertheless, as in Playware, there are questionnaire items for which the agreement between rating and preferences is far from exact (i.e.  $c(z) = 1.0$ ). For instance, correlation values between 0.6 and 0.7, observed in four out of seven question items of the Maze-ball questionnaire, are significant yet raise questions for the several mismatch instances present in the reports.

The second effect suggests that hypothesis H2 is ruled out. The analysis of order of reporting shows, in general, higher order test statistic values in rating than in preferences and significant order effects in four emotional states when reported via a rating scheme. Both indicate a potential higher degree of randomness reporting with rating schemes for that case study.

Finally, note that the consistency of preferences indicated by the  $r_c$  statistic is more often significant for the 2-AFC answers derived from 4-AFC protocol, which is to be expected since 4-AFC explicitly accounts for cases of non-preference.

## 5 Discussion

This initial set of game case studies and the results obtained raise several questions with respect to the relationship between rating and preference self-reports and the particular game survey studies used to test our hypothesis. While a comparison between the two studies is not appropriate given their large set of dissimilarities, an initial analysis across both test cases will assist the design of additional user survey studies that could shed more light to self-reporting effects.

Most significant is the observation that while direct and derived preferences are generally well-correlated, mismatches occur rather frequently and rating questionnaires appear more susceptible to order-of-play effects than preference questionnaires. It is, therefore, interesting to ask *why reported preferences and ratings do not match exactly?* The two studies presented in this paper link the reporting order effect and the existence of randomness in reporting with the inconsistency between the two self-report schemes. The effect of play order is present in most user states examined. Unsurprisingly, these effects vary across different studies, questionnaire schemes and affective states. In both studies there is a general trend of preference for the second game played (recency or order effect) with significant effects appearing only in the rating scheme. Moreover, the statistic measuring the degree of rating consistency suggests that randomness existent

in rating reports appears to be a critical factor for the inconsistency between the two reporting schemes. Preliminary results of a fairer calculation of the  $r_c$  values — including the equal option of preference and allowing for the equality of rating reports — show that consistency is significant only in the preference scheme, which suggests a benefit of preferences for accurate subjective affective reporting and annotation.

A number of other points are worth noting; a study taking account of all of them exceeds the scope of the present paper, but the results reported here suggest that such a study may be worthwhile.

*The experimental protocol favors expressed rating score.* Rating questions were asked **twice as often** as preference questions were asked. Thus, subjects are expected to be familiar with the structure of the rating scheme more than the preference scheme. The preference scheme is arguably **simpler** for the respondent, but requires **increased short-term memory** since at least two — instead of one in the rating scheme — interaction sessions are necessary for comparison. Moreover, the rating scheme question comes first, straight after the experience, followed by the preference scheme. One would, therefore, expect that cognitive and short-term memory load and furthermore questionnaire completion times would be higher when preferences are reported. However, preliminary results from current game survey studies suggest that the time taken to complete a rating questionnaire is significantly higher than a preference questionnaire.

*Questionnaire usability.* Clearly, usability does not affect the results between the two report schemes since the interaction is the same for both: pen-n-paper in Playware, digital bullet-form questionnaire in Maze-Ball.

*Amount of perceived information.* The amount of information provided through the questionnaire is quite unlikely to have an effect on the findings. All questions, preference or rating, are asked in a similar fashion with very small differences — e.g. “I felt challenged” (rating) vs. “I felt more challenged in:” (preferences). The rating schemes used, however, have more available choice options than the preference schemes. For Playware, the options were 2 for preference and 20 for rating. On the other end, rating and preferences have 5 and 4 options, respectively, for Maze-Ball. The thermometer-like rating scheme of Playware appears to generate higher consistencies between preferences and rating but those consistencies are not apparent in all user expressed states of the Maze-Ball study. The thermometer type of rating questionnaire and the 5-option game experience questionnaire (GEQ) [8] are used for their popularity in user and player experience research. A dedicated control experiment is required to explore the impact of the number of options of the questionnaire schemes on the consistency between expressed rating and preference. Four or three-option rating questionnaires could possibly lead to reduced cognitive load of users and higher consistencies.

*Self-report limitations.* Well known limitations of self-reporting such as **self-deception, high intrusiveness and learnability effects** are applicable to both questionnaire schemes and, thereby, do not seem to have a particular impact on the comparison. While there is no clear way to identify such effects, controlling the order of games and questionnaire sessions, as proposed, alleviates in part such effects inherent in naive questionnaires. Other multimodal input sources, including biofeedback and additional context-based game metrics, could be used for further analysis but do not supplant the self-reports.

## Acknowledgments

The authors would like to thank all subjects that participated in the experiments. Special thanks also goes to Héctor P. Martínez for his help in conducting the Maze-Ball user survey experiment. The research was supported, in part, by the FP7 ICT project SIREN (project no: 258453).

## References

1. Agresti, A.: Analysis of ordinal paired comparison data. *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 41(2), 287–297 (1992)
2. Chan, J.C.: Response-order effects in Likert-type scales. *Educational and Psychological Measurement* 51(3), 531–540 (1991)
3. Cohen, J.: A coefficient of agreement for nominal scales. *Educational and Psychological Measurement* 20, 37–46 (1960)
4. Cronbach, J.L.: Coefficient alpha and the internal structure of tests. *Psychometrika* 16(3), 297–334 (1951)
5. Likert, R.: A technique for the measurement of attitudes. *Archives of Psychology* 140, 1–55 (1932)
6. Linn, R., Gronlund, N.: Measurement and assessment in teaching. Prentice-Hall (2000)
7. Lund, H.H., Klitbo, T., Jessen, C.: Playware technology for physically activating play. *Artificial Life and Robotics Journal* 9(4), 165–174 (2005)
8. Poels, K., IJsselsteijn, W.: Development and validation of the game experience questionnaire. In: FUGA Workshop mini-symposium. Helsinki, Finland (2008)
9. Read, J., MacFarlane, S., Cassey, C.: Endurability, engagement and expectations. In: Proceedings of International Conference for Interaction Design and Children (2002)
10. Scheffe, H.: An analysis of variance for paired comparisons. *Journal of the American Statistical Association* 47(259), 381–400 (1952)
11. Stevens, S.S.: On the Theory of Scales of Measurement. *Science* 103(2684), 677–680 (1946)
12. Tognetti, S., Garbarino, M., Bonarini, A., Matteucci, M.: Modeling enjoyment preference from physiological responses in a car racing game. In: Proceedings of the IEEE Conference on Computational Intelligence and Games. pp. 321–328. Copenhagen, Denmark (18–21 August 2010)
13. Viswanathan, M.: Measurement of individual differences in preference for numerical information. *Journal of Applied Psychology* 78(5), 741–752
14. Yannakakis, G.N.: Preference Learning for Affective Modeling. In: Proceedings of the Int. Conf. on Affective Computing and Intelligent Interaction. pp. 126–131. IEEE, Amsterdam, The Netherlands (September 2009)
15. Yannakakis, G.N., Hallam, J.: Towards Optimizing Entertainment in Computer Games. *Applied Artificial Intelligence* 21, 933–971 (2007)
16. Yannakakis, G.N., Hallam, J., Lund, H.H.: Entertainment Capture through Heart Rate Activity in Physical Interactive Playgrounds. *User Modeling and User-Adapted Interaction, Special Issue: Affective Modeling and Adaptation* 18(1-2), 207–243 (February 2008)
17. Yannakakis, G.N., Maragoudakis, M., Hallam, J.: Preference Learning for Cognitive Modeling: A Case Study on Entertainment Preferences. *IEEE Systems, Man and Cybernetics; Part A: Systems and Humans* 39(6), 1165–1175 (November 2009)
18. Yannakakis, G.N., Martínez, H.P., Jhala, A.: Towards Affective Camera Control in Games. *User Modeling and User-Adapted Interaction* 20(4), 313–340 (2010)

# Ratings are overrated!

Georgios N. Yannakakis\* and Héctor P. Martínez

Institute of Digital Games, University of Malta, Msida, Malta

## OPEN ACCESS

**Edited by:**

Javier Jaen,  
Universitat Politecnica de Valencia,  
Spain

**Reviewed by:**

Andreas Duenser,  
Commonwealth Scientific and  
Industrial Research Organisation,  
Australia

Eran Toch,  
Tel Aviv University, Israel  
Donald Glowinski,  
University of Geneva, Switzerland

**\*Correspondence:**

Georgios N. Yannakakis,  
Institute of Digital Games, University  
of Malta, Msida 2080, Malta  
georgios.yannakakis@um.edu.mt

**Specialty section:**

This article was submitted to  
Human-Media Interaction, a section  
of the journal Frontiers in ICT

**Received:** 01 April 2015

**Accepted:** 09 July 2015

**Published:** 30 July 2015

**Citation:**

Yannakakis GN and Martínez HP  
(2015) Ratings are overrated!  
Front. ICT 2:13.  
doi: 10.3389/fict.2015.00013

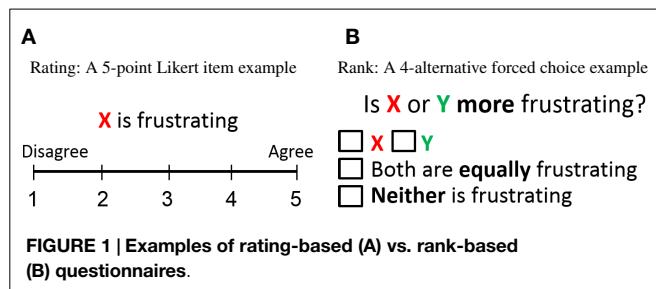
Are ratings of any use in human–computer interaction and user studies at large? If ratings are of limited use, is there a better alternative for quantitative subjective assessment? Beyond the intrinsic shortcomings of human reporting, there are a number of supplementary limitations and fundamental methodological flaws associated with *rating-based questionnaires* – i.e., questionnaires that ask participants to rate their level of agreement with a given statement, such as a Likert item. While the effect of these pitfalls has been largely downplayed, recent findings from diverse areas of study question the reliability of using ratings. *Rank-based questionnaires* – i.e., questionnaires that ask participants to rank two or more options – appear as the evident alternative that not only eliminates the core limitations of ratings but also simplifies the use of sound methodologies that yield more reliable models of the underlying reported construct: user emotion, preference, or opinion. This paper solicits recent findings from various disciplines interlinked with psychometrics and offers a quick guide for the use, processing, and analysis of rank-based questionnaires for the unique advantages they offer. The paper challenges the traditional state-of-practice in human–computer interaction and psychometrics directly contributing toward a paradigm shift in subjective reporting.

**Keywords:** ratings, Likert-scale, ranks, psychometrics, subjective reporting, questionnaires

## Introduction

The key research question within psychometrics and user studies is how to best approximate a user's notion of a subjective construct, such as an experience, a cognitive state, an emotion, or a preference. Even though the *ground truth* of a user's internal state can be manifested via numerous cognitive processes or bodily alterations, it is still far from trivial how to best assess and process those manifestations; entire research areas, such as user experience, user modeling, and affective computing, are long dedicated to this task. Although subjective reporting (first- or third-person) comes with several limitations, such as self-deception and memory-biases, it offers the most direct and popular approach to the annotation of subjective constructs. Thus, quantitative reports via questionnaires provide unique properties for evaluating the capacity of interactive systems (Bardram et al., 2013; Chen et al., 2014) and for constructing computational models of reported user states (Hernandez et al., 2014).

The dominant practice within *human–computer interaction (HCI)* for quantitatively assessing aspects of a user's behavior, experience, opinion, or emotion relies on subjective assessment via rating-based questionnaires – see Bardram et al. (2013), Bryan et al. (2014), Chen et al. (2014), Goyal et al. (2014), Hernandez et al. (2014), Mauderer et al. (2014), Schild et al. (2014), and Sonderegger et al. (2014) among many. Indicatively, a thorough analysis of the papers published in the most prestigious HCI conference last year (Proceedings of CHI'14) reveals that the majority of accepted papers use some form of quantitative assessment approach and more than 80% of these rely on rating-based questionnaires. Popular rating-based questionnaires (see **Figure 1A** for an example) include the Likert-scale (Likert, 1932), the Geneva Wheel model (Scherer, 2005), the Self-Assessment



Manikin (Morris, 1995), the Positive and Negative Affect Schedule (Sonderegger et al., 2014), and the Game Experience Questionnaire (IJsselsteijn et al., 2008). The obtained answers are either used as a means to evaluate an interactive system via the experience of its users – see Bryan et al. (2014), Chen et al. (2014), and Mauderer et al. (2014) – or as data for building predictive models of user reports – i.e., user modeling (Martínez et al., 2014; Hernandez et al., 2014). On the other hand, rank-based questionnaires – which ask the participant to rank a preference between two (or among more than two) options – still remain a rarely used instrument of subjective assessment and modeling, even though there is already significant evidence for their advantages over rating-based questionnaires (Yannakakis and Hallam, 2011; Metallinou and Narayanan, 2013; Čopić Pucihar et al., 2014). An example of a rank-based questionnaire (4-alternative forced choice) is illustrated in **Figure 1B**.

This paper contributes toward a *shift* of the current state-of-practice in user experience, HCI, and psychometrics research at large. For that purpose, the paper provides clear evidence that rating-based evaluation (and modeling) is detrimental to psychometrics and HCI research efforts as it points to biased representations of a user's subjective report. As a result, rating-based instruments are not only of questionable use for the analysis of a subject's report but also evidently lead to unreliable models of those subjects and their reports.

The paper is novel in that it collectively solicits empirical evidence from various research fields, such as marketing research, applied statistics, affective computing, user modeling, and user experience to draw the multiple advantages of rank-based questionnaires for psychometrics and HCI research. At the same time, it provides a comprehensive guide on the use, processing, and analysis of rank-based questionnaires. Toward that aim, we object the use of ratings for HCI based on a number of fundamental limitations and practice flaws (see next section) and we provide empirical evidence for the advantages of ranks (compared to ratings) with respect to subjectivity, order, and inconsistency effects. Furthermore, we suggest appropriate data processing techniques on how to treat ratings – when those are available – and we introduce an open-source toolbox that supports those techniques.

## Ratings: Limitations and Fundamental Flaws

The vast majority of user and psychometric studies have adopted rating questionnaires to capture the opinions, preferences, and perceived experiences of experiment participants – see Bryan et al. (2014), Chen et al. (2014), and Mauderer et al. (2014) among

many. The most popular rating-based questionnaire follows the principles of a Likert-scale (Likert, 1932) in which users are asked to specify their level of agreement with (or disagreement against) a given statement. Ratings have been used, for instance, to report the level of comfort and ease of use of new interfaces or devices (Chen et al., 2014; Weigel et al., 2014) or the stress level during a given task – e.g., in Bardram et al. (2013) and Hernandez et al. (2014). Rating-based reporting, however, has notable inherent limitations that are often overlooked, resulting in fundamentally flawed analyses (Jamieson, 2004). This section sheds some light on the most critical of these limitations and flaws.

### Inherent Limitation: Inter-Personal Differences

Traditionally, HCI studies analyze ratings by comparing their values across participants – see Goyal et al. (2014) and Mark et al. (2014) among many. This is a generally accepted and dominant practice in the community but it neglects the existence of inter-personal differences on the rating process as the meaning of each level on a rating scale may differ across experiment participants. For example, two participants may assess the exact same level of “ease to use” for a new device but then one rates it as “very easy to use” and the other as “extremely easy to use.” There are numerous factors that contribute to the different internal rating scales existent across participants (Metallinou and Narayanan, 2013), such as differences in personality, culture (Sneddon et al., 2011), temperament, and interests (Viswanathan, 1993). As these factors are documented extensively in the literature, the appropriateness of the dominant HCI state-of-practice is directly questioned.

A large volume of studies have also identified the presence of primacy and recency order effects in rating-based questionnaires e.g., Chan (1991) and Yannakakis and Hallam (2011), seen as systematic biases toward parts of the scale (Linn and Gronlund, 2000) (e.g., right handed participants may tend to use the right side of the scale) or a fixed tendency over time (e.g., on a series of experimental conditions, the last ones are rated higher). Indicatively, the comparative study of Yannakakis and Hallam (2011) between ratings and ranks showcases higher inconsistency effects and significant order (recency) effects existent in ratings across two different datasets, which contain both rank and rating annotations obtained from the same participants. Although these are systematic biases (opposed to personal), they pose additional challenges on the comparison of ratings among participants, as participants are affected to different extents. Even though experiments on quantifying human perception through rating questionnaires have led to interesting findings on the relationship between perception and reporting, biases of the use of ratings as an assessment tool have not been examined (Jay et al., 2007).

### Ratings are Not Numbers

In addition to inter-personal differences, a critical limitation arises when ratings are treated as interval values since ratings are by nature ordinal values (Stevens, 1946; Jamieson, 2004). As a result, any method that treats them as numbers (e.g., average values, *t*-tests, linear models) is fundamentally flawed. In most questionnaires, Likert items are represented as pictures [e.g., different representations of arousal in the Self-Assessment Manikin (Morris, 1995)] or as adjectives (e.g., “moderately,” “fairly,” and “extremely”). These labels (images or adjectives) are

often erroneously converted to integer numbers violating basic axioms of statistics, which suggest that ordinal values cannot be treated as interval values (Stevens, 1946) since the underlying numerical scale is unknown. Note that even when a questionnaire features ratings as numbers (e.g., see **Figure 1A**), the scale is still ordinal as the numbers in the instrument are only labels; thus, the underlying numerical scale is still unknown and dependent on the participant (Stevens, 1946; Langley and Sheppeard, 1985; Ovadia, 2004). Moreover, when treated as numbers, equal ratings are considered of equal value. This is another invalid assumption to make as questionnaires do not always provide sufficient granularity. By treating ratings as ordinal values, this issue is avoided as only the relations among unequal values are considered.

## The Non-Linearity of Ratings

Treating ratings as interval values is grounded in the assumption that the difference between consecutive ratings is fixed (i.e., ratings follow a linear scale). However, there is no valid assumption suggesting that a subjective rating scale is linear (Jamieson, 2004). For instance, the difference between “fairly (4)” and “extremely (5)” may be larger than the distance between “moderately (3)” and “fairly (4)” as some experiment participants rarely use the extremes of the scale or tend to use one extreme more than the other (Langley and Sheppeard, 1985). If, instead, ratings are treated naturally as ordinal data no assumptions are made about the distance between rating labels, which eliminates introducing flawed information and data noise to the analysis.

## Why Should I Use Ranks Instead?

A rank-based questionnaire scheme asks experiment participants to compare and sort a number of options. On its simplest form, the participants compare two options and specify which one is the preferred under a given statement (pairwise preference). For instance, participants could select which of two devices is easier to use. With more than two options, the participants are asked to provide a ranking of some or all the options. At a remote observation, one may argue that ranks provide less information than ratings as they do not express a quantity explicitly and only provide ordinal relations. As argued in the previous section, however, any additional information obtained by ratings when treated as numbers violates basic axioms of applied statistics. Thus, ratings do not provide data for a richer analysis if appropriately treated as ordinal values.

Being a form of subjective reporting rank-based questionnaires (as much as rating-based questionnaires) is associated with well known limitations, such as memory effects and self-deception. Reporting about subjective constructs, such as experience, preference, or emotion via rank-based questionnaires, however, has recently attracted the interest of researchers in marketing (Dhar and Simonson, 2003), psychology (Brown and Maydeu-Olivares, 2013), user modeling (Yang and Chen, 2011; Baveye et al., 2013), and affective computing (Tognetti et al., 2010; Martínez et al., 2014) among other fields. This gradual paradigm shift is driven by both the reported benefits of ranks minimizing the effects of self-reporting subjectivity and recent findings demonstrating the advantages of ranks over ratings. Inspired by the seminal work of Scheffe (1952) and Agresti (1992) for

the analysis of paired comparisons Yannakakis and Hallam (2011) compared data from rating and rank-based questionnaires across a number of domains and identified increased order and inconsistency effects when ratings are used. Evidence from findings by Metallinou and Narayanan (2013) also suggest that rank-based annotation of emotion should be preferred to rating-based annotation for its ability to eliminate annotation biases (cultural, subjective, inconsistency, inter-rater, etc.).

In summary, results across different domains investigating subjective assessment suggest that rank-based reports minimize the assumptions made about experiment participants’ notions of highly subjective constructs, such as experience and emotions, and allow a fair comparison among the answers of different participants. Moreover, artifacts, such as the subjective notion of scaling, are eliminated. Finally, all these advantages also lead to the construction of generalizable and accurate computational models of users or their experience (Martínez et al., 2014).

## What if Ratings is All I Have?

The core findings from the areas of applied statistics, user modeling, affective computing, machine learning, and marketing research discussed already not only suggest that ranks define a superior instrument for subjective assessment but they also *question the very use* of ratings at the first place. One could, however, still claim that the use of ratings in some particular experimental protocols is unavoidable. For instance, in experimental protocols, subjects can only be asked to assess their experience on solely one version of an interactive system (e.g., a game, a web-browser).

When faced with such a condition ratings could provide a viable assessment instrument if they are naturally treated as ordinal data. A recent study by Martínez et al. (2014) investigates the effect of using ratings as nominal or ordinal scales when studying the relation between physiological attributes and emotional states. Both approaches are tested on synthetic (testing “*in vitro*”) and human (testing “*in vivo*”) ratings. The core findings of the study across all datasets examined provide clear evidence that ratings (when used) should be naturally transformed to ordinal representations (ranks). This practice has clear benefits: any data analysis followed yields more reliable and generalizable outcomes as those better approximate the underlying *ground truth* of the reported subjective construct. The transformation from ratings to ranks is straightforward. Ratings are compared to one another and a pairwise preference/ranking is created for every pair/tuple; higher ratings take the top positions of the ranking and lower ratings the positions in the bottom of the ranking. Comparisons of ratings from different experiment participants must be avoided. In addition, if the time window between reported ratings is sufficiently large for the examined task (e.g., in the magnitude of hours or more) one can also consider removing particular rating pairs to reduce artifacts connected to participants’ episodic memory.

In general, approaches for analyzing ordinal ratings should rely on *non-parametric* statistics: from simple statistical explorations via *Spearman’s* correlation, to significance tests via the *Mann–Whitney* test and the *Wilcoxon signed-rank* test for paired samples (Wilcoxon, 1945), to the *Kruskal–Wallis* (Kruskal and Wallis, 1952) and *Friedman’s* (Friedman, 1940) tests for three (or more) groups of ranks. Clearly, statistical models, such as

artificial neural networks and support vector machines, are also suitable for the analysis of ordinal data (Martínez et al., 2014).

Norman (2010) showed empirically in one dataset that *Pearson's* correlation (which treats ratings as intervals) is robust enough when compared against *Spearman's* rank correlation (which treats ratings as ordinal values). Such evidence could support the validity of using standard *parametric* correlation tests that treat ratings as interval values but does not question the very use of ratings due to their inherent limitations. On the contrary, a number of studies have demonstrated the supremacy of ranks in eliminating various forms of reporting biases [e.g., Yannakakis and Hallam (2011)]. Finally, significant improvements have been reported in accuracies of non-linear statistical models when ratings are treated as ordinal values (Martínez et al., 2014).

## How to Analyze Ranks

Standard data visualization methods based on averages or SDs are strictly not applicable on ordinal data – obtained directly as ranks or transformed from ratings. Instead, to explore the relationships between ranks and a number of considered factors, a stacked bar chart can be used to visualize how many observations were assigned to each rank for each value of the factor.

For a statistical factor analysis, a common choice is the *Wilcoxon signed-rank test* (Wilcoxon, 1945), sometimes also used to evaluate the effect of ratings as, for instance, in Mauderer et al. (2014). This is a paired-samples test and, therefore, guarantees that only within-participant ranks are compared, bypassing inter-personal differences. A common alternative is *Kendall's Tau* (Kendall, 1938) that can be used to calculate the correlation between the hypothesized order (e.g., device A is easier to use than device B) and the observed ranks – see, e.g., Martínez et al. (2014). The non-parametric *Kruskal–Wallis* and *Friedman's* tests mentioned earlier are also applicable.

Furthermore, if an HCI researcher is interested in using the reported ranks to build computational models that predict those ranks (e.g., constructing models of users) a large palette of algorithms is currently available. Linear statistical models, such as linear discriminant analysis and large margins, and non-linear approaches, such as Gaussian processes, artificial neural networks, and support vector machines, are applicable for learning to predict ranks. These methods are derived from the sub-area of machine learning named *preference learning* (Fürnkranz and Hüllermeier, 2010). A number of such preference learning methods as well as data preprocessing and feature selection algorithms are currently included in the *preference learning toolbox (PLT)* (Farrugia et al., 2015). PLT is an open-access, user-friendly, and accessible toolkit<sup>1</sup> built and constantly updated for the purpose of easing the processing of (and promoting the use of) ranks.

## Summary of Conclusions

This paper directly objects to the use and analysis of subjective assessment via ratings within quantitative user studies, HCI and psychometrics research contributing to a shift from the dominant state of practice in those fields. Beyond any well-reported

limitations of subjective reporting (e.g., memory effects, self-deception) ratings come with inherited limitations as an instrument of reporting. These are derived from inter-personal differences and include, among many, high-inconsistency effects and subjectivity of scaling. Those effects question the very use of ratings for obtaining valid data for any further analysis. Most importantly, the traditional analysis of ratings within HCI and psychometrics – i.e., deriving statistical properties from ratings, such as average and variance values – violates two fundamental assumptions. The first common flaw is the violation of the assumption that ratings are ordinal data. The second assumption violated is that ratings evidently are not linear (even if they could be represented as numbers). In response to the above mathematical violations, in principle, ratings *should not* be converted to numerical scales and analyzed as numbers.

Rank-based questionnaires are the alternative instrument for subjective quantitative assessment proposed in this paper. Recent findings from a number of fields including applied statistics, affective computing, user modeling, and machine learning provide clear evidence for the supremacy of ranks, when compared to ratings, on minimizing subjectivity biases as well as order, inter-rater, and inconsistency effects (Yannakakis and Hallam, 2011; Metallinou and Narayanan, 2013). More so, recent evidence suggests that we can construct more accurate and reliable statistical models of reported ratings – that better approximate the underlying ground truth of the subjective construct we attempt to measure – only when ratings are naturally treated as ordinal data (Martínez et al., 2014).

Given the supremacy or rank-based subjective assessment for both evaluating interactive systems (through the experience of their users) and as the ground truth for deriving computational models of subjective reports, this paper serves as a guide for both rank-based evaluation and rank-based computational modeling. For the former, it provides methods for the conversion of ratings to ranks – when ratings are available – and an overview of statistical processes for rank reports. For the latter, it proposes preference learning methods and algorithms – incorporated to an open-source, accessible toolbox – for the construction of predictive models of ranks.

It is important to stress that this paper did not intend to present *yet another* case study to further prove empirically the advantages of ranks over ratings or demonstrate the general flaws of processing ratings. Our claims are not based on the popularity of ranks in other fields outside HCI, but on empirical findings as surveyed in the paper. While the limitations of ratings and ranks have been identified and discussed extensively, no other study within the HCI community both solicits evidence for the comparative advantages of ranks (as demonstrated in other fields) and offers a short guidebook on how to process ranks statistically. We hope that this paper highlights the obvious fundamental issues of ratings as a subjective assessment tool and introduces ranks as the alternative reporting approach toward altering a dominant, yet falsified, community practice.

## Acknowledgments

The work is supported, in part, by the EU-funded FP7 ICT ILearnRW project (project no: 318803).

<sup>1</sup><http://sourceforge.net/projects/pl-toolbox/>

## References

- Agresti, A. (1992). Analysis of ordinal paired comparison data. *J. R. Stat. Soc. Ser. C Appl. Stat.*, 41, 287–297.
- Bardram, J. E., Frost, M., Szántó, K., Faurholt-Jepsen, M., Vinberg, M., and Kessing, L. V. (2013). Designing mobile health technology for bipolar disorder: a field trial of the monarca system. *Proc. SIGCHI Conf. Hum. Factor. Comput. Syst.* 2627–2636. doi:10.1145/2470654.2481364
- Baveye, Y., Bettinelli, J. N., Dellandrea, E., Chen, L., and Chamaret, C. (2013). A large video database for computational models of induced emotion. *Proc. of Affect. Comput. Intell. Int.* 13–18. doi:10.1109/ACII.2013.9
- Brown, A., and Maydeu-Olivares, A. (2013). How irt can solve problems of ipsative data in forced-choice questionnaires. *Psychol. Methods* 18, 36. doi:10.1037/a0030641
- Bryan, N. J., Mysore, G. J., and Wang, G. (2014). Isse: an interactive source separation editor. *Proc. SIGCHI Conf. Hum. Factor. Comput. Syst.* 257–266. doi:10.1145/2556288.2557253
- Chan, J. C. (1991). Response-order effects in Likert-type scales. *Educ. Psychol. Meas.* 51, 531–540. doi:10.1177/0013164491513002
- Chen, X. A., Grossman, T., Wigdor, D. J., and Fitzmaurice, G. (2014). Duet: exploring joint interactions on a smart phone and a smart watch. *Proc. SIGCHI Conf. Hum. Factor. Comput. Syst.* 159–168. doi:10.1145/2556288.2556955
- Čopić Pucihar, K., Coulton, P., and Alexander, J. (2014). The use of surrounding visual context in handheld ar: device vs. user perspective rendering. *Proc. SIGCHI Conf. Hum. Factor. Comput. Syst.* 197–206. doi:10.1145/2556288.2557125
- Dhar, R., and Simonson, I. (2003). The effect of forced choice on choice. *J. Market. Res.* 40, 146–160. doi:10.1509/jmkr.40.2.146.19229
- Farrugia, V. E., Martínez, H. P., and Yannakakis, G. N. (2015). The preference learning toolbox. [arXiv:1506.01709].
- Friedman, M. (1940). A comparison of alternative tests of significance for the problem of m rankings. *Ann. Math. Stat.* 11, 86–92. doi:10.1214/aoms/1177731944
- Fürnkranz, J., and Hüllermeier, E. (2010). *Preference Learning*. New York: Springer.
- Goyal, N., Leshed, G., Cosley, D., and Fussell, S. R. (2014). Effects of implicit sharing in collaborative analysis. *Proc. SIGCHI Conf. Hum. Factor. Comput. Syst.* 129–138. doi:10.1145/2556288.2557229
- Hernandez, J., Paredes, P., Roseway, A., and Czerwinski, M. (2014). Under pressure: sensing stress of computer users. *Proc. SIGCHI Conf. Hum. Factor. Comput. Syst.* 51–60. doi:10.1145/2556288.2557165
- IJsselsteijn, W., Poels, K., and De Kort, Y. (2008). *The Game Experience Questionnaire: Development of a Self-Report Measure to Assess Player Experiences of Digital Games*. Eindhoven: TU Eindhoven.
- Jamieson, S. (2004). Likert scales: how to (ab) use them. *Med. Educ.* 38, 1217–1218. doi:10.1111/j.1365-2929.2004.02012.x
- Jay, C., Glencross, M., and Hubbeld, R. (2007). Modeling the effects of delayed haptic and visual feedback in a collaborative virtual environment. *ACM Trans. Comput. Hum. Interact.* 14, 1275514. doi:10.1145/1275511
- Kendall, M. G. (1938). A new measure of rank correlation. *Biometrika* 81–93. doi:10.1093/biomet/30.1-2.81
- Kruskal, W. H., and Wallis, W. A. (1952). Use of ranks in one-criterion variance analysis. *J. Am. Stat. Assoc.* 47, 583–621. doi:10.1080/01621459.1952.10483441
- Langley, G., and Shepphard, H. (1985). The visual analogue scale: its use in pain measurement. *Rheumatol. Int.* 5, 145–148. doi:10.1007/BF00541514
- Likert, R. (1932). A technique for the measurement of attitudes. *Arch. Psychol.* 14, 1–55.
- Linn, R., and Gronlund, N. (2000). *Measurement and Assessment in Teaching*. Upper Saddle River, NJ: Prentice-Hall.
- Mark, G., Wang, Y., and Niiya, M. (2014). Stress and multitasking in everyday college life: an empirical study of online activity. *Proc. SIGCHI Conf. Hum. Factor. Comput. Syst.* 41–50. doi:10.1145/2556288.2557361
- Martínez, H. P., Yannakakis, G. N., and Hallam, J. (2014). Don't classify ratings of affect; rank them! *IEEE Trans. Affect. Comput.* 5, 314–326. doi:10.1109/TAFFC.2014.2352268
- Mauderer, M., Conte, S., Nacenta, M. A., and Vishwanath, D. (2014). Depth perception with gaze-contingent depth of field. *Proc. SIGCHI Conf. Hum. Factor. Comput. Syst.* 217–226. doi:10.1145/2556288.2557089
- Metallinou, A., and Narayanan, S. (2013). “Annotation and processing of continuous emotional attributes: challenges and opportunities,” in *Automatic Face and Gesture Recognition (FG), 2013 10th IEEE International Conference and Workshops on* (Shanghai: IEEE), 1–8.
- Morris, J. (1995). Observations: sam: the self-assessment Manikinan efficient cross-cultural measurement of emotional response. *J. Advert. Res.* 35, 63–68.
- Norman, G. (2010). Likert scales, levels of measurement and the laws of statistics. *Adv. Health Sci. Educ.* 15, 625–632. doi:10.1007/s10459-010-9222-y
- Ovadia, S. (2004). Ratings and rankings: reconsidering the structure of values and their measurement. *Int. J. Soc. Res. Method.* 7, 403–414. doi:10.1080/1364557032000081654
- Scheffe, H. (1952). An analysis of variance for paired comparisons. *J. Am. Stat. Assoc.* 47, 381–400. doi:10.2307/2281310
- Scherer, K. (2005). What are emotions? and how can they be measured? *Soc. Sci. Inform.* 44, 695–729. doi:10.1177/0539018405058216
- Schild, J., La Viola, J. J. Jr., and Masuch, M. (2014). Altering gameplay behavior using stereoscopic 3d vision-based video game design. *Proc. SIGCHI Conf. Hum. Factor. Comput. Syst.* 207–216. doi:10.1145/2556288.2557283
- Sneddon, I., McKeown, G., McRorie, M., and Vukicevic, T. (2011). Cross-cultural patterns in dynamic ratings of positive and negative natural emotional behaviour. *PLoS ONE* 6:e14679. doi:10.1371/journal.pone.0014679
- Sonderegger, A., Uebelbacher, A., Pugliese, M., and Sauer, J. (2014). “The influence of aesthetics in usability testing: the case of dual-domain products,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY: ACM), 21–30.
- Stevens, S. S. (1946). On the theory of scales of measurement. *Science* 103, 677–680. doi:10.1126/science.103.2684.677
- Tognetti, S., Garbarino, M., Bonarini, A., and Matteucci, M. (2010). “Modeling enjoyment preference from physiological responses in a car racing game,” *Computational Intelligence and Games (CIG), 2010 IEEE Symposium on* (Copenhagen: IEEE), 321–328.
- Viswanathan, M. (1993). Measurement of individual differences in preference for numerical information. *J. Appl. Psychol.* 78, 741–752. doi:10.1037/0021-9010.78.5.741
- Weigel, M., Mehta, V., and Steimle, J. (2014). More than touch: understanding how people use skin as an input surface for mobile computing. *Proc. SIGCHI Conf. Hum. Factor. Comput. Syst.* 179–188. doi:10.1145/2556288.2557239
- Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics Bull.* 80–83. doi:10.2307/3001968
- Yang, Y. H., and Chen, H. H. (2011). Ranking-based emotion recognition for music organization and retrieval. *IEEE Trans. Audio Speech Lang. Process.* 19, 762–774. doi:10.1109/TASL.2010.2064164
- Yannakakis, G. N., and Hallam, J. (2011). Rating vs. preference: a comparative study of self-reporting. *Proc. Affect. Comput. Intell. Int.* 6974, 437–446. doi:10.1007/978-3-642-24600-5\_47
- Conflict of Interest Statement:** The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.
- Copyright © 2015 Yannakakis and Martínez. This is an open-access article distributed under the terms of the Creative Commons Attribution License (CC BY). The use, distribution or reproduction in other forums is permitted, provided the original author(s) or licensor are credited and that the original publication in this journal is cited, in accordance with accepted academic practice. No use, distribution or reproduction is permitted which does not comply with these terms.*

# Emotionally Adapted Games – An Example of a First Person Shooter

Timo Saari<sup>1</sup>, Marko Turpeinen<sup>2</sup>, Kai Kuikkanemi<sup>2</sup>, Ilkka Kosunen<sup>2</sup>,  
and Niklas Ravaja<sup>3</sup>

<sup>1</sup> Temple University, 1801 N. Broad Street, Philadelphia, PA, USA, and Center for Knowledge and Innovation Research (CKIR), Helsinki School of Economics, Finland and Helsinki Institute for Information Technology (HIIT), Finland  
saari@temple.edu

<sup>2</sup> Helsinki Institute for Information Technology (HIIT), Finland  
{Marko.turpeinen, kai.kuikkanemi, ilkka.kosunen}@hiit.fi

<sup>3</sup> Center for Knowledge and Innovation Research (CKIR),  
Helsinki School of Economics, Finland  
Niklas.Ravaja@hse.fi

**Abstract.** This paper discusses a specific customization technology – Psychological Customization - which enables the customization of information presented on a computer-based system in real-time and its application to manipulating emotions when playing computer games. The possibilities of customizing different elements of games to manipulate emotions are presented and a definition of emotionally adaptive games is given. A psychophysiological adaptive game is discussed as an example of emotionally adapted games.

**Keywords:** Customization, adaptive systems, psychological effects, emotion, games, emotionally adapted games, psychophysiological measurement, Psychological Customization.

## 1 Introduction

Emotions or emotion-related variables (e.g., competitiveness) play a critical role in gaming behavior [1, 2]. People seek, and are eager to pay for, games that elicit positive emotional experiences and enjoyment; however, an enjoyable game may not elicit only positive emotions but possibly also negative ones (e.g., anger, fear). Thus, one of the major goals for video game designers is to elicit optimal emotional responses or response patterns.

It is possible to build systems which either automatically or semi-automatically adapt games to create optimal emotional responses and response patterns. Such customization systems entail the manipulation of the game per player to elicit specific emotional responses or to create desired response patterns. We propose a system called Psychological Customization to be used in the adaptation of games for manipulating emotional states of players.

Psychological Customization entails the customization of *transient* (i.e. short-term) *user experiences* (i.e. psychological effects) when interacting with media- and communication technologies. Experience-based customization entails the automatic or semi-automatic adaptation of information per user, task and context in an intelligent way with information technology. A subset of Psychological Customization is to vary the form of information (modality for instance) per user profile, task and context, which may systematically manipulate (approach, avoid, modify intensity, frequency and duration, create combinations, create links to behavior) different psychological effects. Psychological effects can be considered transient states, such as emotion, mood, types of cognition, learning, flow, presence, involvement and enjoyment. [e.g. 3, 13]

Psychological Customization works on the principle of target experiences which can be set by using the system either by providers of a service or by users. Target experiences are different types of transient psychological states that have varying durations, frequencies, intensities, combinations, and motivational and action tendencies as well as a linked stimulus class which facilitates a particular state. The system is set up to either approach or avoid a certain target experience within the other parameters of customization such as altering the intensity, duration or frequency of a certain effect or creating simultaneous combinations or links to probable behavior of different target experiences. [3]

Psychological Customization can infer customer needs via user models and various feedback loops observing customer behavior and responses. Psychological Customization also provides different adaptations of presenting information to different customers based on the customer interacting with the configuration settings of the product or service.

The basic functioning of the system is based on a classic control theory model, the biocybernetic loop. It defines two kinds of control loops in complex and adaptive systems that can be established: negative (avoid an undesirable standard) and positive (approach a desirable standard) loops of feedback [e.g. 4, 5]. Target experiences are then controlled by this type of reasoning in the system based on real-time feedback from user responses and/or based on ready-made design-rule databases.

**Emotion.** Although various definitions of emotions have been proposed, the most general definition is that emotions are biologically based action dispositions that have an important role in the determination of behavior [e.g., 6 ]. It is generally agreed that emotions comprise three components: subjective experience (e.g., feeling joyous), expressive behavior (e.g., smiling), and physiological activation [e.g., sympathetic arousal, 7].

There are two main competing views of emotions. Proponents of the basic distinct emotions argue that emotions, such as anger, fear, sadness, happiness, disgust, and surprise, are present from birth, have distinct adaptive value, and differ in important aspects, such as appraisal, antecedent events, behavioral response, physiology, etc. [8]. In contrast, according to a dimensional theory of emotion, emotions are fundamentally similar in most respects, differing only in terms of one or more dimensions. Proponents of the dimensional view have suggested that all emotions can be located in a two-dimensional space, as coordinates of valence and arousal [or bodily activation; e.g., 6, 9]. The valence dimension reflects the degree to which an affective experience is negative (unpleasant) or positive (pleasant). The arousal

dimension indicates the level of activation associated with the emotional experience, and ranges from very excited or energized at one extreme to very calm or sleepy at the other.

Other theorists have, however, suggested that the two main, orthogonal dimensions of emotional experience are negative activation (NA) and positive activation (PA) that represent a 45° rotation of the valence and arousal axes [10]. The NA axis extends from highly arousing negative emotion (e.g., fear) on one end to low-arousal positive emotion (e.g., pleasant relaxation) on the other, while the PA axis extends from highly arousing positive emotion (e.g., joy) to low-arousal negative emotion (e.g., depressed affect). The self-report NA and PA dimensions have been suggested to represent the subjective components of the BIS and BAS, respectively [e.g., 10, 11].

We adopt the latter definition of emotion. On the NA axis we call the high arousal negative emotion anxiety and stress while the low arousal emotion can be termed as pleasant relaxation. On the PA axis we see the high arousal emotion as joy and the low arousal emotion as depression.

In our studies we have successfully used both psychophysiological measurements and self-report to index emotional processes, also when playing computer games. In computer games, there is a dynamic flow of events and action, games potentially eliciting a multitude of different emotions varying across time. A serious limitation of prior game studies is that they have used tonic, rather than phasic, psychophysiological measures. Tonic measures (e.g., the mean physiological value during the game minus pre-game baseline) do not enable the examination of the varying emotions elicited by different instantaneous game events. Given that psychophysiological measurements can be performed continuously with a high temporal resolution, it is possible to quantify phasic responses to instantaneous game events (e.g., by comparing the local pre-event baseline to physiological activity immediately following event onset). [12]

It is then evident that psychophysiological measurement of emotional states when playing a computer game is both feasible and fruitful in providing an account of some aspects of the moment-to-moment experience of the user [13]. Naturally, psychophysiology could be extended to function as a feedback loop into the gaming engine making real-time adaptation of the game relative to the emotional state or mood of the user a possibility.

## 2 Emotionally Adapted Games

Emotionally adapted gaming can be seen as based on gaming templates which are parts of the meta-narrative of the game. Hence, a basic approach to an element to be adapted inside a game is a psychologically validated template which creates a particular psychological effect. A broad view of templates may be that the whole game consists of a database of psychologically validated templates that are dynamically presented by the gaming engine in sequences during gameplay. A limited view entails that a smaller collection of templates is used. The element of psychological evaluation means that the selected psychological influence (such an emotional response) of the template on a particular type of user is sufficiently predictable. These psychologically evaluated templates may consist of i) manipulating

the substance of a game, such as story line (initiating events, new characters etc.) and manipulating the situations specifically related to the character of the player (such as putting the character into sudden and dangerous situations inside the game) and ii) manipulating the form or way of presentation of the game (such as visual elements, shapes, colors, types of objects, sound effects, background music, level of interactivity and feedback etc.). The difficulty level of the game may also be continuously automatically be adjusted, thereby keeping the skills and challenges in balance, which results in a maintenance of an optimal emotional experience and possibly also a flow-state. [14]

Why and when then to manipulate emotion in gaming on the basis of avoiding or approaching a specific emotional state? First, there are the transient basic emotional effects of games that are dependent of the phase of the game or some specific events. These are emotions such as happiness, satisfaction, sadness, dissatisfaction, anger, aggression, fear and anxiousness. These emotions are the basis of narrative experiences, i.e. being afraid of the enemy in a shooting game, feeling aggression and wishing to destroy the enemy and feeling satisfaction, even happiness, when the enemy has been destroyed. Emotional regulation systems in these instances most naturally may focus on manipulating the event structures, such as characters, their roles, events that take place and other features of the narrative gaming experience. [14]

Second, there are possibilities for emotional management, especially in the case of managing arousal, alertness and excitation. Also, one may wish to manage negative emotions, such as sadness, dissatisfaction, disappointment, anger, aggression, fear and anxiousness. The case for managing these emotions is twofold. On the one hand, one may see that these emotions could be eliminated altogether in the gaming experience. This can happen via either eliminating, if possible, the emergence of such an emotion in the game. For example, one can make a deliberately happy game with level-playing monkeys in a far away island throwing barrels at obstacles and gathering points. This would include minimum negative emotions. Or, in a game where negative emotion is a basic part of the game, one may wish to limit the intensity, duration or frequency of the emotions via manipulating gaming events and gaming elements so that sadness or fear are at their minimum levels, or that gaming events do not lead to sadness at all. [14]

Similarly, managing level of arousal or the intensity, duration and frequency of select negative emotions may be quite feasible in the case of children as a form of parental control. On the other hand, one may wish to maximize arousal, alertness and excitation, perhaps even anger, fear and aggression for hardcore gamers.

Third, there are possibilities related to the avoidance of certain types of emotions that are typically indicative of a poor gaming experience. Inactivity, idleness, passivity, tiredness, boredom, dullness, helplessness as well as a totally neutral experience may be indicating that there is some fundamental problem in the user-game interaction. This could be due to poor gaming skills of the user vs. the difficult challenges of the game or some other factors, such as the user is stuck in an adventure game for too long and can not proceed without finding a magic key to enter the next level or so. When a gaming engine detects these emotions in the user, it may adapt its behavior to offer the user more choices of selecting the difficulty level of the game or offer the user some clues as to how to go forward in the game. The game can also adapt its level of difficulty to the player's skill level. [e.g. 14]

Fourth, it is also possible to create different combinations of emotional states (satisfied and angry) or emotional states and other psychological states (pleasant relaxation and efficient information processing) or emotional states and behavior (using specific motivational and action tendencies). [3]

All of these possibilities may be relevant. However, the elimination or minimization of certain emotions may be specifically feasible in the case of indicated overly poor gaming experience in which the game may adapt its behavior to assist the user. It should be noted that events in games may change quickly and produce complex situations and hence complex emotions that may change rapidly. Consequently, one should better integrate these approaches into the genre or type of the game, such as driving simulator, first person shooter, sports game such as golf, or an adventure game, or a level-playing game for children. [14]

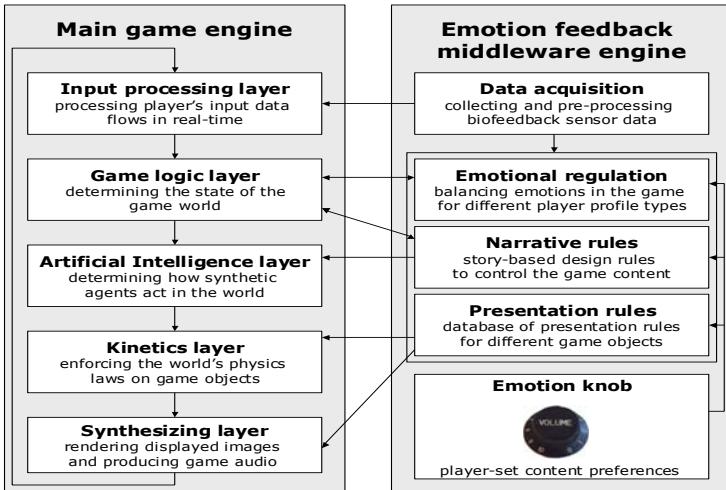
### **3 Example: Psychophysiological Adaptive First-Person Shooter Game**

We will now present a basic system schematic of an emotionally adapted game in Figure 1. The process of a typical gaming engine is depicted on the left-hand side of the diagram. The engine continuously monitors user input, which is typically collected using a keyboard, a joystick, or other game controllers. This input data is then processed and transferred to the layer that handles the game's internal logical state, and the user input may influence the game state. After the logical state of the game is defined the system alters the actions of the synthetic agents in the game world. For example, these include the actions of computer-controlled non-player characters. The complexity of this AI layer varies greatly depending on the game. Based on the game state and the determined actions of the synthetic agents, the physics engine determines the kinetic movements of different objects within the world. Finally, the game world is synthesized for the player by rendering the graphical elements and producing and controlling the audio elements within the game. [see 14]

The proposed emotional regulation can be implemented as a middleware system that runs parallel to the actual game engine. The input processing layer of the game engine can receive a data flow of captured and pre-processed sensor data. The real-time signal processing may consist of different forms of amplifying, filtering and feature selection on the psychophysiological signals. This data flow may directly influence the state of the game world, or it can be used by the emotional regulation sub-module of the emotion feedback engine. This module consists of the rules of emotional balancing for different player profile types and gamer-related explicitly set preferences controlled by the “emotion knob”. In addition, it contains a collection of design rules for narrative constructions and game object presentation within the game world. The emotional regulation module also receives input from the game engine's logical layer to make selections related to desired emotional balance and narrative structures within the game. [14]

The outputs of emotional regulation engine may then be applied to various different levels of the actions of the game engine: i) the logical state of the world may be re-directed, ii) the actions of the synthetic agents may be controlled, iii) the

kinetics of the game may be altered and iv) the rendering of the game world may be changed. First two options are more relevant to high-level and story-related structures of the game, whereas the last two are more directly related to the selection of presentation of objects within the virtual environment. [e.g. 14]



**Fig. 1.** Emotional adaptation system design for games. Adapted from [14].

With our system design for games it is possible for the game designer as well for the user to set desired emotional targets to be approached or avoided. The system uses both positive and negative feedback loops to determine the ideal adaptations case-by-case for game play for various emotional effects to be realized and managed.

Indeed, to implement and evaluate some of the ideas presented, we have explored novel technical solutions and tested different kinds of psychophysiological adaptations that can be implemented. EMOShooter is a prototype platform for psychophysiological adaptive 3D first-person shooter (FPS) gaming. It is built on top of open-source graphics engine (OGRE 3D) and physics engine (ODE). In this experimental platform we have the possibility to modify practically any game world element, player avatar, avatar outlook, or control parameter.

EMOShooter is a simple psychophysiological adaptive game and hence a part of our emotionally adapted games definition. The system uses psychophysiological signals to influence the ease of use of the controls of the game hence affecting game play difficulty and game play experience. The system does not have target experiences systematically implemented at this moment nor does it have an emotion knob to tune the system. However, the EMOShooter game is a valuable example of one type of emotionally adapted games in demonstrating one feasible link between real-time emotional state measurement with psychophysiology and the game play.

The goal of the EMOShooter game is to kill cube-like enemies either with sniper or machine gun. We have been testing various adaptation patterns with EMOShooter by primarily EDA and respiration as psychophysiological signals in our adaptive

feedback system regards how these signals can be meaningfully connected to the actual game play via adapting game controls.

Adaptation of game controls includes changes in rate of fire, recoil, movement speed and shaking. If a player is aroused this will be reflected in EDA and respiration signals which in turn will make rate of fire and movement slower and will make the aim shaky. Hence, for a highly aroused player the game becomes more difficult. For a mildly aroused or calm player the controls become more efficient and easy to use hence facilitating performance at game play. Game events are mostly arousing. The amount of cubes to shoot, their approach and firing on the user, the amount of health left after being hit and the sound effects all are geared to drive up arousal in the game. The player's task is to be calm as indexed by psychophysiological signals to be able to operate the controls more efficiently.

In our tests of the game we have collected also EMG data to infer the valence dimension of emotion during game play. In addition to the psychophysiological signals we have collected data from the players using behavioral game logging, video capture, interviews and questionnaires. During our tests we noticed that proper calibration and base lining of the psychophysiological signals is very important for the adaptations to work. We also noticed that having robust stimuli in the game is crucial for the adaptations to work because in many cases the stimulus functioned as a trigger in adaptation. The psychophysiological signals used are calibrated by using dynamic range (basically a variation of dynamic signal normalization algorithm), which has a memory buffer of a few seconds (depending on signal). Dynamic range is easy to use and effective calibration mechanism, and relative change seems to be more practical than absolute values in this kind of gaming.

According to our early analysis, there are three key issues in designing psychophysiologically adaptive games i) understanding the meaningful emotionally adaptive gaming patterns, ii) implementation of adaptation algorithms and signal processing, and iii) purposeful use of sensors in the game context [15].

The design patterns used in emotionally adaptive gaming must be meaningful and enjoyable for the player, and the utilization of signals must also obey the overall goal of the game. In order to achieve the goal player should find the right rhythm or balance of playing the game and control of psychophysiological responses and signals.

Signals should be analyzed as close to real-time as possible in psychophysiologically adaptive gaming in order to keep the feedback loop in pace with the game adaptations and game events. We have used time-series analysis with short sample windows. In practice, ECG, EEG and EMG always require extensive data processing, but EDA and respiration can be almost used as such to create the adaptation signal. This implies that not all psychophysiological signals are equally open to be used as real-time inputs into an adaptive game at least in this stage of signal processing hardware and software development.

Usability of psychophysiological recording devices remains quite poor. Respiration, HR [heart rate] and EDA are probably the easiest to implement. Also in case of emotional adaptation the design of the game may include the physical design of the sensors, e.g. "Detective hat" for EEG sensors or "Sniper-gloves" for EDA sensors. Hence, the sensors could be designed as part of the game story rather than presented as cumbersome and invasive laboratory-originated equipment.

In future versions of EMOShooter we may also employ the system design of emotionally adapted games including setting of explicit experiential targets and their parameters for gaming sessions and the emotion control knob.

## 4 Discussion

Gaming, as we have presented it in this article, is perhaps one of the most promising application areas of Psychological Customization. We see that both casual and hardcore gamers could benefit from the use of our system and entirely new types of games can be created. Psychological Customization would enable game designers to use our tools both when developing the game and testing it with users in a rapid manner as well as part of the final product. From the user's point of view, using various types of control knobs of emotion or other experiences enables them to customize and have more control over their gaming experiences.

Good games are composed of delicate synthesis of the components creating a pleasant game balance and challenge for players. Introducing emotional adaptation increases the complexity of game design tasks involved. However, regards the economics of game development our system would not induce a dramatic cost. The system automatically establishes gaming patterns and structures which would fill a target experience and its parameters. Our system could be a modular toolset that can be adapted to various types of gaming platforms and gaming engines. The emotional tuning knob could be integrated into existing game controls including sliders for level of graphic violence in the game, for instance. Of course, development work is needed to create an easy-to-use game adaptation interface for users to set their preferences for game play.

There are several challenges for Psychological Customization systems in games. We see three main areas which are critical: i) measurement of experience, ii) quality of reasoning in an adaptive system and iii) acceptability by users.

The measurement of experience in the first place is a key challenge. However, we have presented an approach to concentrate on those aspects of experience, such as emotion, which are perhaps better defined than experiential states in general. Despite this focus, there is still disagreement in theorizing, operationalizing and measuring emotions.

The quality of reasoning in an adaptive system is often a bottle-neck in performance. If a closed system is produced with fixed design-rules it would inevitably encounter situations, stimuli and users which would challenge the systems fixed rules and produce errors in adaptations. While there are several deep and sophisticated technical and mathematical approaches to this problem including different ways of machine learning and reasoning (which are beyond the scope of this article), we propose a higher-level solution possibility. Our answer is to rely on the co-evolutionary potential of our system design with changing user models and emergent design-rules. Of course, this approach needs working algorithms and techniques to form the necessary metadata and other data structures to be processed by the system.

User acceptance of invasive psychophysiological measurement as input to the game is critical. Hardcore gamers may be more suspect to accept new peripheral

devices linking them to game than gaming novices or casual gamers. However, the culture of connecting one's body to a game is already evolving. Think of **Wii** as an example with a controller tied to one's wrist, constantly touching the skin. It would not be unimaginable to think of psychophysiological sensors embedded in similar controls as people are more used to "semi-invasive" gaming controls beyond the use of a mouse and keyboard. The solution here could be to design sensors as **embedded** into essential existing or new types of gaming peripherals. A driving wheel with EDA and ECG sensors or driving gloves with similar sensors with added blood pressure, muscle tension and finger movement sensors could be used as easily acceptable controls of a driving simulator, for example.

In our tests of the psychophysiologically adaptive game as a first prototype of emotionally adapted games we have been able to produce meaningful gaming patterns and game adaptations. We argue that our approach to emotionally adapted games is novel and creates new opportunities for designing games. We feel that our approach may result in a new type of enabling technological platform focused on the customization of gaming experiences. This new enabling technology platform can facilitate the development new types of games but can also be used with existing types of games and gaming platforms.

## References

1. Grodal, T.: Video games and the pleasures of control. In: Zillmann, D., Vorderer, P. (eds.) *Media entertainment: The psychology of its appeal*, pp. 197–212. Lawrence Erlbaum Associates, Mahwah (2000)
2. Vorderer, P., Hartmann, T., Klimmt, C.: Explaining the enjoyment of playing video games: The role of competition. In: Marinelli, D. (ed.) *Proceedings of the 2nd International Conference on Entertainment Computing (ICEC 2003)*, Pittsburgh, pp. 1–8. ACM Press, New York (2003)
3. Saari, T., Turpeinen, M., Ravaja, N.: Technological and Psychological Fundamentals of Psychological Customization Systems – An Example of Emotionally Adapted Games (A manuscript submitted for publication)
4. Pope, A.T., Bogart, E.H., Bartolome, D.S.: Biocybernetic system evaluates indices of operator engagement in automated task. *Biological Psychology* 40, 187–195 (1995)
5. Wiener, N.: *Cybernetics: Control and communication in the animal and the machine*, 2nd edn. MIT Press, Cambridge (1948)
6. Lang, P.J.: The emotion probe. Studies of motivation and attention. *American Psychologist* 50, 372–385 (1995)
7. Scherer, K.R.: Neuroscience projections to current debates in emotion psychology. *Cognition and Emotion* 7, 1–41 (1993)
8. Ekman, P.: An argument for basic emotions. *Cognition and Emotion* 6, 169–200 (1992)
9. Larsen, R.J., Diener, E.: Promises and problems with the circumplex model of emotion. In: Clark, M. (ed.) *Review of personality and social psychology*, vol. 13, pp. 25–59. Sage, Newbury Park (1992)
10. Watson, D., Wiese, D., Vaidya, J., Tellegen, A.: The two general activation systems of affect: Structural findings, evolutionary considerations, and psychobiological evidence. *Journal of Personality and Social Psychology* 76, 820–838 (1999)

11. Gray, J.A.: The neuropsychology of temperament. In: Strelau, J., Angleitner, A. (eds.) *Explorations in temperament: International perspectives on theory and measurement*, pp. 105–128. Plenum Press, New York (1991)
12. Ravaja, N., Laarni, J., Saari, T., Kallinen, K., Salminen, M.: Phasic Psychophysiological Responses to Video Game Events: New Criterion Variables for Game Design. In: *Proceedings of HCI 2005*, Las Vegas, NV, USA (2005)
13. Saari, T., Turpeinen, M.: Towards Psychological Customization of Information for Individuals and Social Groups. In: Karat, J., Blom, J., Karat, M.-C. (eds.) *Personalization of User Experiences for eCommerce*. Kluwer, Dordrecht (2004)
14. Saari, T., Ravaja, N., Turpeinen, M., Kallinen, K.: Emotional Regulation System for Emotionally Adapted Games. In: *Proceedings of FuturePlay 2005 conference*, Michigan State University, USA, October 13-15 (2005)
15. Kuikkanemi, K., Laitinen, T., Kosunen, I.: Designing emotionally adaptive gaming. In: *Proceedings of EHTI 2008: The First Finnish Symposium on Emotions and Human-Technology Interaction*, University of Tampere, Tampere, Finland, pp. 13–18 (2008)



## Dynamic difficulty adjustment on MOBA games

Mirna Paula Silva\*, Victor do Nascimento Silva, Luiz Chaimowicz

*Department of Computer Science, Federal University of Minas Gerais (UFMG), Belo Horizonte, Brazil*

### ARTICLE INFO

#### Article history:

Received 3 February 2016  
Revised 15 September 2016  
Accepted 3 October 2016  
Available online 4 October 2016

#### Keywords:

Artificial intelligence  
Digital games  
Dynamic difficulty adjustment  
Dynamic difficulty balance  
Entertainment  
MOBA

### ABSTRACT

This paper addresses the dynamic difficulty adjustment on MOBA games as a way to improve the players entertainment. Although MOBA is currently one of the most played genres around the world, it is known as a game that offer less autonomy, more challenges and consequently more frustration. Due to these characteristics, the use of a mechanism that performs the difficulty balance dynamically seems to be an interesting alternative to minimize and/or avoid that players experience such frustrations. In this sense, this paper presents a dynamic difficulty adjustment mechanism for MOBA games. The main idea is to create a computer controlled opponent that adapts dynamically to the player performance, trying to offer to the player a better game experience. This is done by evaluating the performance of the player using a metric based on some game features and switching the difficulty of the opponent's artificial intelligence behavior accordingly. Quantitative and qualitative experiments were performed and the results showed that the system is capable of adapting dynamically to the opponent's skills. In spite of that, the qualitative experiments with users showed that the player's expertise has a greater influence on the perception of the difficulty level and dynamic adaptation.

© 2016 Elsevier B.V. All rights reserved.

### 1. Introduction

The game industry is growing at a fast pace, globally generating more revenue than film and music industries [41]. Games are considered a great source of entertainment [30] and, due to that, the industry is increasingly investing more resources in research and development. This allows developers to create realistic graphics, deep narratives and complex artificial intelligence (AI), leading to games even closer to reality [25,37].

The development of realistic games results in an improved player immersion which, in general, increases their satisfaction [4]. Although this is a well explored approach, it is not the only way to make games more attractive. According to Yannakakis and Hallam [49], the player's psychological factor makes direct influence to this attractiveness, requiring the game to maintain the player interested on it. An approach to captivate the player into the game experience is to make the challenges directly associated to the player's skill [11]. However, a game may not suit the expectation of players with different skills. While a player may have a hard time in final levels of a game, there may be another player that cannot win the initial ones. This scenario requires that the game dynamically adjusts itself presenting challenges that suit the needs and skills of each player. This game adjustment can be

performed by a technique called dynamic difficulty adjustment (DDA) or dynamic difficulty balancing.

In spite of different studies in DDA [39,38,42,2,46], none of them tackles MOBA (Multiplayer Online Battle Arena) games, which are one of the most played games genres nowadays, having almost 30% of online computing gameplay time.<sup>1</sup> Although they are very popular among gamers, there is not much attention from researchers over this game genre. This can be related to the inherent challenges of developing competitive artificial intelligent agents for MOBA games as well as the constant updates and changes on these games.

This paper presents a dynamic difficulty adjustment mechanism for MOBA games. The main idea is to create a computer controlled opponent that adapts dynamically to the player performance, trying to offer to the player a better game experience. This is done by evaluating the performance of the player using a metric based on some game features and switching the difficulty of the opponent's artificial intelligence behavior accordingly. This idea was initially proposed in [34,33], and here we revisit the mechanism, giving more details about its implementation and performing a set of experiments with human players in order to have a qualitative evaluation. We also present and discuss the main characteristics and challenges of MOBA games, trying to encourage other researchers to use them as testbeds in their future work.

\* Corresponding author.

E-mail address: [mirnancia@gmail.com](mailto:mirnancia@gmail.com) (M.P. Silva).

<sup>1</sup> <http://goo.gl/zgKjJL>.

This paper is organized as follows: in Section 2 we present the related work and background on difficulty balance; Section 3 covers MOBA games aspects, history and challenges, as well as the game DotA used as testbed in this work; Section 4 addresses the methodology and the proposed mechanism; Section 5 discusses the performed experiments of agents versus agents and the obtained results; Section 6 presents the experiments performed with users and the collected results; and finally, Section 7 brings the conclusion and directions for future work.

## 2. Difficulty balance

Difficulty balance, or difficulty adjustment, consists on doing modifications to parameters, scenarios and/or game behaviors in order to avoid the player's frustration when facing the game challenges [11,22]. According to Mateas [27] and Hunnicke [17], it is possible to adjust all game features using the correct algorithms, from storytelling to maps and level layouts, all online. These adjustments allow the game to adapt itself to each player, making he/she entertained throughout the game. To make this possible, Andrade et al. [1] describes that the dynamic difficulty adjustment must attend three basic requirements. First of all, the game must automatically identify the players' skills and adapt to it as fast as possible. Second, the game must track the player's improvement and regressions, as the game must keep balance according to the player's skill. At last, the adaptive process must not be explicitly perceived by players, keeping game states coherent to previous ones. However, before applying the dynamic difficulty adjustment, it is necessary to understand the meaning of difficulty.

The meaning of difficulty is abstract in many ways and some aspects should be taken into account to evaluate and measure difficulty. For this measuring, we can consider level design characteristics [3], amount of resource or enemies [17], amount of victories or losses [32,47], among other metrics. Nevertheless, dynamic difficulty adjustment is not as simple as just giving player additional health items when in trouble. This problem requires estimation of time and intervention in the right moment, since maintaining the player entertained is a complex task in an interactive context [17].

A wide range of tasks and challenge levels can be found in games. For example, tasks that require high skill and synchronism (First Person Games), tasks that require logic and problem solving skills (Puzzles), tasks related to planning (Strategy games), and so on [21]. According to Klimmt et al. [21], there is evidence that the completion of tasks and challenge overcoming are directly related to player satisfaction and fun. Yannakakis [48] developed a study about the most popular approaches for player modeling during interaction with entertainment systems. According to this study, most qualitative approaches proposed for player entertainment modeling tends to be based in conceptual definitions proposed by Malone [26] and Csikszentmihalyi [8].

Malone [26] defended the need for a specific motivation during gameplay to entertain the player. The necessary features to reach such motivation are: *fantasy, control, challenges and curiosity*. The use of *fantasy* as part of game world could improve player motivation, creating objects, scenarios or situations that the player could explore. *Control* is a player feeling through which he/she is part of game control. Given the interaction of games, all of them make the player feel involved in game control and the control levels can change from game to game. *Challenge* implies that the game should pursue tasks and goals in an adequate level, making the player feel challenged to his/her limits. The uncertainty of completing tasks or goals provided by game mechanics encourages the player motivation. Finally, *curiosity* suggests that game information must be complex and unknown, to encourage exploration and reorganization of information by players. Games must pursue multiple situa-

tions or scenarios from the main course since it helps to stimulate the player to explore the unknown [26,13].

The qualitative approach proposed by Csikszentmihalyi [8] is called *flow theory* or *flow model*. According to the author, *flow* is a mental state experienced when the user is executing an activity in which he/she is immersed, feeling focused, completely involved and fulfilled during task execution. So, this model takes into account the psychological steps that players reach during gameplay. In this sense, the main goal is controlling the challenge levels aiming to maintain the player inside the flow, avoiding to reach boredom (no challenges at all) or frustration (challenges are too hard). Fig. 1 show a graph of flow theory presented by Csikszentmihalyi [8].

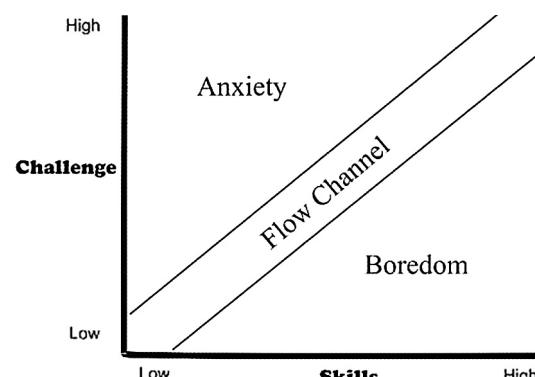
The model presented by Csikszentmihalyi shows how a task difficulty is directly related to the perception of who is executing it. The flow channel illustrates that difficulty can be progressively improved, since there exists time to the player to learn and improve his/her skills to overcome this challenge [9]. Thereby, this model avoids frustration of very hard situations or boredom caused by very easy situations. Furthermore, Csikszentmihalyi and Nakamura [10] go beyond and determine that the ratio of challenges to skills should be around 50/50 in order to produce enjoyable experiences.

On the other hand, there are some studies that question if the ratio of challenges to skills is really a measure of flow. Løvoll and Vittersø [24], for instance, present a work with some empirical evidence that contests the idea that flow is produced when challenges and skills are harmonized. According to them, the interaction between challenges and skills as independent variables does not support the challenge skill ratio proposed by Csikszentmihalyi and Nakamura.

In a different approach, if we can balance the fantasy, control, challenge and curiosity proposed by Malone [26] and associate it to the progressive development of difficulty presented by Csikszentmihalyi [8], it is possible that the resulting game can entertain the player. However, using just these features is not sufficient to show if game challenges are compatible with player skills. So, it is necessary measuring techniques to define when and how difficulty should be adjusted.

### 2.1. Evaluating the difficulty level

According to Andrade et al. [1], there are some different approaches to dynamically balance the difficulty level of a game. However, all of these approaches require measuring, implicitly or explicitly, the difficulty level that the player is facing on that moment. This measurement can be done by using *heuristics*, for example the success rate of skill landing, the capture of enemy



The Flow. After Mihaly Csikszentmihalyi, *The Flow* (1990), p. 74

Fig. 1. Diagram of flow theory, by Csikszentmihalyi.

points, the time used to complete a task or any other metric that can evaluate the player. Missura and Gärtnert [28] made a relation between game runtime, health and score in a way that it composes an evaluation criteria that performs the game difficulty adjustment. Demasi and Adriano [12] developed a heuristic function called “Challenge Function” that is responsible for describing the game state, and tries to show how hard the game is for the player in a given time.

Another way to track difficulty levels is using some physiological signs, informally called **body language**. Van Den Hoogen et al. [43] mentions that the body language of a player could be related to his/her experience during play. According to the authors, there are evidences that show that specific postures, facial expressions, eye movements, stress over mouse/keyboard/joystick, and others, could evidence experiences like interest, excitement, frustration and boredom. For the evaluation of player experience, authors created a monitoring ambient, placing pressure sensors at different devices (mouse, chair, etc.). Also cameras were placed to register movements and facial expression. The results of this experiment show that the behaviors observed are directly related to the excitement level and dominance felt during the game. Nacke and Lindley [29], besides using cameras to capture body language, also used electrodes to track mental reaction from players during a First Person Shooter (FPS) match. The results obtained during player monitoring were based on the flow theory proposed by Csikszentmihalyi [8], therefore, authors could observe if the players were inside the flow, anxious or bored during the gameplay.

Although the explicit measuring (external monitoring) of difficulty levels could provide fine results related to game fitness to player's skill, it is impracticable to the dynamic difficulty adjustment. Not all players have measuring tools at home and using such tools could be intrusive, since this could make the player uncomfortable by being monitored. Implicit approaches (metrics and heuristics) do not need external equipment, therefore these approaches are more popular among game developers. Besides, they contribute to the fact that players must not perceive that difficulty is being adjusted during gameplay.

This paper tries to perform a dynamic difficulty adjustment through the development of a mechanism that switches between three distinct levels of artificial intelligence in order to provide an opponent that better suits the player's abilities. The mechanism performs several evaluations during the match, detecting the moments in which the game is unbalanced, and then executes the difficulty adjustment.

### 3. Multiplayer online battle arena

Multiplayer Online Battle Arena, also known as Action Real-time Strategy, or simply as MOBA, is a genre originated from Real-Time Strategy (RTS), as a modification of the original game. The first known MOBA game is *Aeon of Strife*, created from the game Starcraft. In this game, the player should choose an unit and work his/her way to conquer the enemy's base with the chosen unit and its special powers. This game structure was maintained through the improvement of MOBAs. One interesting fact is that these games came up from simple fan made games to become one of the most played genres in the world, as will be discussed later in this section.

Another interesting fact is that according to Johnson et al. [20] and Kwak et al. [23], MOBA games were found to offer less autonomy, more frustration and more challenges to players. These findings with respect to autonomy seems most likely to be a function of the fact that MOBA games involve fairly focused competition with other players. Moreover, the greater levels of frustration experienced may also be a function of the focused competition that

occurs in MOBA games and the steep learning curve. With less focus on the qualities of the game and greater focus on competing and cooperating with others, there is more potential for frustration with the performance of other players. This interpretation is supported by players reporting a greater challenge when playing MOBA games [36]. Due to these characteristics, the use of a mechanism that performs the difficulty balance dynamically seems to be a viable alternative to minimize and/or avoid that such frustrations be experienced by the players.

In this section we present the MOBA history, its characteristics and gameplay. Then, we discuss the unique features present in DotA, the selected platform to be used as testbed in this work, and why it is so hard to develop AI agents to play against humans on these games.

#### 3.1. History

As Real-Time Strategy (RTS) games became popular, we observed the urge of the players to create their own maps and gameplay styles. This phenomena resulted in a community of developers that later came to be known as *modders*. Their work was known as *mods*, an acronym to the word “modification”. In those mods, players could use the original game environment to play by their own rules, allowing them to create a fantasy world beyond the limits of the original game.

Released in 1998, *Aeon of Strife* was the first mod to present a unique characteristic that caught RTS players attention. Instead of focusing on resource collection and base construction, the mod valued the player ability to control a single unit, an ability known as *micromanagement*. This characteristic invited players to duel against each other into single or teams battles, showing to be a successful approach to get players involved in the game.

*Aeon of Strife* inspired many other mods that followed its guidelines. Later in 2005, one of those mods stood out in the crowd: *Defense of the Ancients* (DotA). The platform was not Starcraft anymore, but another game developed by Blizzard: Warcraft III. The game had the perfect environment and an open API that allowed the modders to do their job. Therefore, they created the DotA map where players would assume the control of a single unit called *Hero* and develop this unit by defeating enemies, just like in a Role Playing Game (RPG). Every hero has a single set of attributes and powers, characterizing them in a role. The player could then choose a hero based on its team needs or on its own gameplay style. The game story of the DotA map were also inherited from the Warcraft myth: the war between two races in the Warcraft world, the Night Elf and the Undead. Thereby, players were invoked to defend a main structure called *Ancient*, which must be destroyed in order to achieve the game goal. The game is divided into two teams with five players each: The Sentinel, having the Tree of Life as *Ancient*; and The Scourge, having the Frozen Throne as *Ancient*. Screen shots of the team bases can be found in Fig. 2.

The DotA popularity among players resulted in behavioral changes in the general gameplay. Instead of just playing on LANs, players were excited about playing on the Internet. At that time, the broadband was expanding all around the world and players wanted to test it, as well as test their skills challenging others around the world. There were platforms like Garena that had dedicated servers only for DotA matches. DotA's gameplay became so famous that it inspired the game industry to create professional games based on this play style. In 2009, Riot Games released a game called *League of Legends* (LoL) [15], with characteristics very similar to DotA. This company created the term MOBA, referring to their debuting title as a Multiplayer Online Battle Arena. Later, Valve has released its own game, known as *Dota2*, that immediately caught the attention of the world and media because of the 1 million Dollar tournament. Lastly, around 2010, S2 games have



**Fig. 2.** The sentinel base (top) and the scourge base (bottom).

released its own title, *Heroes of Newerth*, that has similar characteristics to *DotA*, *Dota2* and *LoL*. Nowadays, there are other titles, such as *Strife* and *Heroes of the Storm*, but they did not get many players as the games released before.

In numbers, we can see that MOBA genre is a world success, as shown in Fig. 3. By 2012, the game *League of Legends* has overcome *World of Warcraft* as the most played game in the world [16]. In November 2015, as reported by Raptr, *League of Legends* alone represented more than 22% of the worldwide gameplay.<sup>2</sup> There are international eSports competitions involving those games and millionaire prizes. In 2015, for example, the official Valve's World Tournament of *Dota2* called "The International" distributed a total of US\$ 18 million.<sup>3</sup>

### 3.2. Gameplay and characteristics

To provide challenges that suit the player's skills it is necessary to comprehend the gameplay that involves the game. The MOBA game can be summarized into two teams playing against each other: Team 1 and Team 2. Players on the Team 1 are based at the southwest corner of the map, and those on the Team 2 are based at the northeast corner. Each base is defended by towers and waves of NPC units (called creeps) that guard the main paths leading to their base, called lanes. In the center of each base there is one main structure. This structure is the goal of the game, which the enemy must destroy in order to win the match.

The teams are composed by five players, where each player controls one specific and powerful unit with unique abilities, which is called Hero or Champion. In most MOBAs, players on each team choose one from dozens of heroes, each with different abilities and tactical advantages over the others. The scenario is highly team-oriented: it is difficult for one player to lead the team to victory by himself/herself.

Since the gameplay goes around strengthening individual heroes, it does not require focus on resource management and base-building, unlike most traditional RTS games. When killing enemy

or neutral units, the player gains experience points and when enough experience is accumulated the player increases his/her level. Leveling up improves the hero's toughness and the damage it inflicts, allowing players to upgrade spells or skills.

In addition to accumulating experience, players also manage a single resource of gold that can be used to buy items such as armory, potions, among others. Besides a small periodic income, heroes can earn gold by killing hostile units, towers, base structures, and enemy heroes. With gold, players can buy items to strengthen their hero and gain abilities. Also, certain items can be combined with recipes to create more powerful items. Buying items that suit the chosen hero is an important tactical element of the game.

### 3.3. Map

The map is segmented into three different lanes, the top, the bottom, and the middle lane. Each one of these lanes leads to the other team's base, guarded by towers along the way. Fig. 4 represents a general MOBA map with its lanes, bases and towers along each lane.

The map area located between the lanes is called jungle. This is where neutral creeps can be found, which can be killed for gathering more gold and experience points. It is possible to level up by killing creeps in the jungle instead of in the lanes. This practice is called jungling.

During the early laning phase of the game, most gameplay is centered around "farming". It means that players focus on collecting resources and leveling up their heroes by defeating enemy units, like creeps or heroes. In the case of the junglers, they walk through the jungle and kill neutral units. Further, the junglers and their team seek for failures on the enemy teams' strategy, looking for catching them in traps or performing gang killing, the so called ganks. Lastly, there is the late game phase, when the gameplay is commonly focused on teamfights, i.e., teams use their heroes to fight in groups, looking for weakening the enemy team and pushing the lane towards the enemy's base.

Each team has defensive towers placed along the lanes leading to the Ancient. Those towers inflict heavy single target damage to heroes and creeps. In the early stages of the game, a hero can only take a few hits from a tower before dying, so one must be careful as to not get in a bad positioning relatively to the towers until they have gained enough strength. In Fig. 4 the towers are represented by little circles placed in the lanes.

### 3.4. Defense of the ancients

The game DEFENSE OF THE ANCIENTS (DotA) is a Multiplayer Online Battle Arena (MOBA) mod version of the game WARCRAFT III: REIGN OF CHAOS and later to its expansion, WARCRAFT III: THE FROZEN THRONE. The scenario objective is for each team to destroy the opponents' Ancient, heavily guarded structures located at opposing corners of the map. Players use powerful units known as heroes, and are assisted by allied heroes (played by other users) and AI-controlled fighters known as creeps. As in role-playing games, players level up their heroes and use gold to buy items and equipment during the match.

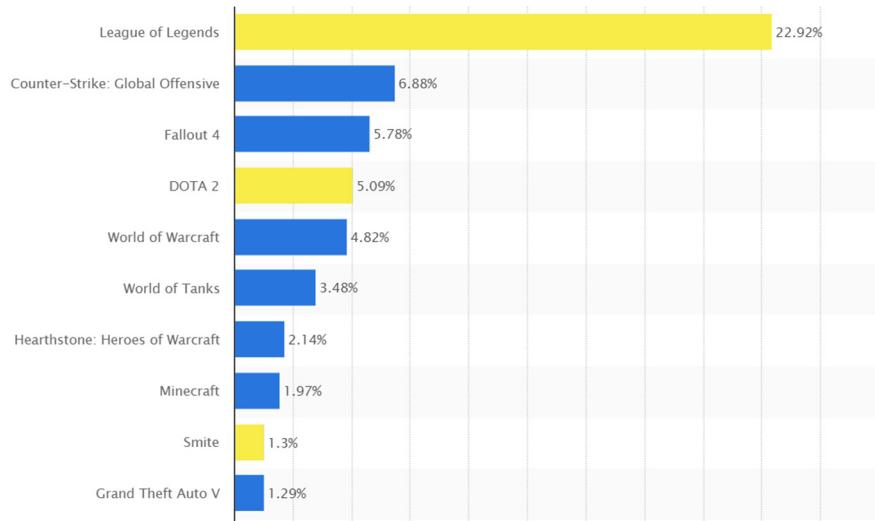
Moreover, since DotA is a mod of the game WARCRAFT III: REIGN OF CHAOS, it becomes easier to modify because we can use the tools made to edit WARCRAFT maps to do it. Therefore, the game DEFENSE OF THE ANCIENTS (DotA) was chosen to be the testbed of this work.

### 3.5. Game adaptations

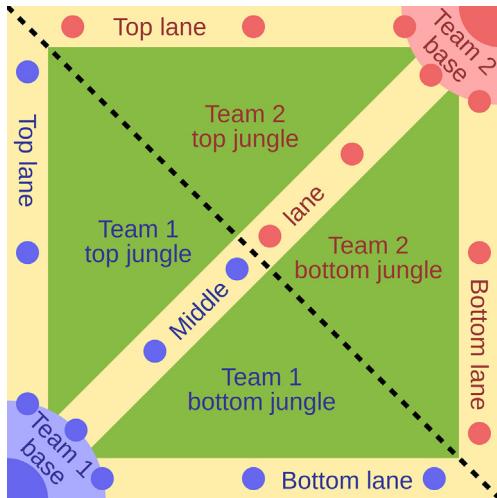
To use the game DEFENSE OF THE ANCIENTS as a testbed, some adaptations were made in order to better suit the needs of this work. The

<sup>2</sup> <http://goo.gl/zgKjJL>.

<sup>3</sup> <http://goo.gl/6IXfMD>.



**Fig. 3.** Ten most played games of 2015. Source: Raptr/Statista.



**Fig. 4.** General map layout from MOBA games.

original game allows the player to choose his/her hero among 110 different options. But, for this work, we chose to restrict this quantity to only 10 heroes, equally distributed between both teams.

Each hero has distinct characteristics, behaviors and abilities. Thereby, to better focus on the strategies and the development of abilities, we designed our artificial intelligence agent to control one specific hero. The selection performed was random and the chosen character is *Lion - The Demon Witch*. Given this choice, it became possible to classify which abilities and behaviors should be implemented so that the artificial intelligence agent would work with a consistent behavior during the game match. Fig. 5 shows a screenshot of the character *Lion - THE DEMON WITCH* during a game match.

The DotA game also offers a variety of game modes, selected by the game host at the beginning of the match. The game modes dictate the difficulty of the scenario, as well as whether people can choose their hero or are assigned one randomly. Many game modes can be combined, allowing more flexible options. In this work we restrict the game mode to single selection, that means that a player does not receive a random hero, but is allowed to select among nine others, because *Lion* is automatically picked by the AI agent.



**Fig. 5.** Screenshot of the hero during a match.

### 3.6. The challenges of developing a MOBA agent

Developing agents capable of defeating competitive human players in MOBA games remains an open research challenge. According to Buro [5] and Weber et al. [44], improving the capabilities of computer opponents would increase the game playing experience and provide several interesting research questions for the artificial intelligence community. However, developing an AI agent to play MOBA games is not a simple task [35]. Although there are several AI agents for all the different MOBA distributions, none of them can perform as well as expert human players. One of the reasons for this is due to the inability of AI systems to learn from experience. Human players only need a couple of matches to identify opponents' weaknesses and use them in their favor in upcoming games. Current machine learning approaches in this area are not good enough when compared to expert humans skills [5,45]. Yet, according to Buro [6], some commercial game AI systems may outperform human players and may even create challenging encounters, but they do not advance our understanding on how to create intelligent entities, since it cheats to compensate its lack of sophistication by using map revealing and faster resource gathering.

Since MOBA games are originated from Real-Time Strategy (RTS) genre, many of the challenges that surround RTS games can also be applied to MOBA. A case study for real-time AI problems in the context of RTS games can be found in Buro [5,6]; Buro and Furtak [7].

As discussed before, MOBA provides a complex environment, populated with dynamic and static features. Moreover, MOBA characteristics tends to make the game more dynamic than its precursor, the RTS genre. Fights, duels, and actions happen in a short time, all requiring the computation of complex algorithms to analyze the scenario and to reason about it. For instance, team-fights normally last a few seconds and the agent has to perform a large amount of computation in a short time, to reason about allies, enemies, and strategies. Even for humans, it is difficult to maintain the total control of the situation during these fights.

Although not having the macromanagement that occurs in RTS games, MOBA matches require the player to reason about thousands of combinations of spells and items. The spell leveling order, item buying and building order matter, because each spell and each item has its own characteristics, making a special ability that highlights the hero early in the game. Such combinations should take into account the enemy that is being faced by the agent, the opponent team in general, the items and combinations from its own team, among many other features in the game. Even more, those features are not always clear to be translated in a language that can be easily understood by the agent, since it requires experience and sometimes knowledge that goes beyond the game itself.

Being a commercial game genre, MOBA provides a rich hero pool, allowing the player to choose among hundreds of heroes. Performing combinations of heroes on the team can lead to success or defeat even in the hero picking phase of the game. Selecting the right hero to be played against another hero, or a set of heroes that can face the opponent's set is a difficult task. This choice requires knowledge about the teammates' heroes, the development curve, the hero classification and trying to predict the enemy's team strategy. Moreover, each hero in a MOBA game is designed with a role. That means that a hero will be better developed if it is played in the role that it was designed. Picking the right heroes for the right roles requires all the knowledge cited above, and it is a hard task for the AI agent, since it requires knowledge that goes beyond of the game scope, commonly denominated *metagame*. For instance, in a situation where the team composed by five weak, low-damage dealers are fighting against a team composed by five strong, high-damage dealers sounds like a bad choice, since the first team will struggle on all battles against the opponent during the match.

Lastly, MOBA games, as RTS, provide a partially observable environment. Dealing with the uncertainty of this situation is hard for most agents, because it requires sophisticated motion planning algorithms, and real-time reasoning about the environment. There are some MOBAs, like *Heroes of the Storm*, that even integrate a *bush* in the game scenario, providing spots where the player cannot be seen if his/her hero is inside a bush. This allows players to perform a wide range of tactical plays, like traps, faking and ganking. Reasoning about these fast-paced plays is not trivial, and therefore, requires predictions and especial research efforts.

#### 4. Methodology

Our difficulty adjustment mechanism consists in the development of three different levels of artificial intelligence that will be chosen during the match in order to present challenges that suit the player's skills. To select the right opponent, a difficulty evaluation is performed during the game and if it indicates that the players are not evolving in the same pace, it executes the necessary adjustment. Throughout this section, we address the artificial

intelligence agent developed, the game features, the difficulty evaluation process, and the mechanism to dynamically adjust the presented difficulty during a match.

##### 4.1. Artificial intelligence agent

To be able to provide an opponent that can face different skilled players, the artificial intelligence agent must be implemented with distinct ability levels to simulate the most different behaviors played. Since the agent must simulate an opponent player, the developed algorithm implements actions and behaviors to a hero unit. During a game match, this hero should follow the player's performance, so if the player is having a good evolution, the hero controlled by artificial intelligence must be able to also do the same. However, if the player is not evolving enough or if his/her development start to decrease, the AI agent that controls the hero must lower its pace and keep up with its opponent.

The hero behavior was divided into three categories: easy mode, regular mode and hard mode. Each one of these categories has singular aspects that aim to be suitable to players with different abilities. These are described below.

###### 4.1.1. Easy mode

In the easy mode, the hero performs regular attacks every time an enemy enters in its attack range. When an allied tower is under attack, the hero detects the need for defense and moves towards the attacked ally in order to defend it. Another strategic action is how the hero chooses the enemy tower to be its main target. Every time the hero starts a moving action, it analyses which of the enemy's towers has taken more damage and is closer to be defeated. Once it finds, the hero sets that tower as the main target and goes in that direction. It is important to mention that, in the easy mode, all the attacking actions that the hero performs are basic attacks. The hero also retreats as a defense strategy. So when its health points are below 30%, it starts to retreat towards its base, where it can recover its health when it reaches a specific recovery building. The easy mode was created for beginners or some less skilled players, where the implemented strategies are not very complex and do not use any special character skill (also known as spells).

###### 4.1.2. Regular mode

In the regular mode, besides the strategies implemented for the easy mode, the hero also starts to manipulate items. The item manipulation is very helpful to improve the hero's attributes and also to recover some attributes that have been decreased, for example, items to recover health points or mana. Likewise, there are items to increase attributes like strength, speed, intelligence, among others. As part of the defense strategy, if the hero's health points reach 30% or less, it will first use some health potion to recover it and if these items are over, then the hero starts to retreat towards its base. The regular mode was created to match those players that have already some experience and know how to use some of the game functionalities in his/her favor but are not experts yet.

###### 4.1.3. Hard mode

The hard mode has all the strategies implemented on both preceding modes, besides its own specific actions. Here, the hero goes beyond item manipulation and starts to learn, improve and cast spells. Spells are unique skills that each hero has. These spells can give a more effective damage on the enemy, can boost the recovery of its own attributes (like mana or health points), can give some kind of advantage to allied units (like freezing the enemies), among other possibilities. Every time the hero gains a new level it also gains one attribute point to distribute among its spells. So in

this mode, besides the regular attack, the hero also casts spells to attack enemies or defend allies. Here we also decided to implement a new strategy for a head-to-head combat. In order to avoid losing the combat against another hero, the artificial intelligence agent algorithm keeps monitoring the area around its hero. Therefore, if an enemy hero enters the monitored area, the hero controlled by the agent will take advantage on that and will begin to attack it. The strategies to defend allied towers and to retreat are the same developed on regular mode. The hard mode was created to match those players that have more experience on the DotA game and also know how to use the game functionalities in their favor. This kind of player may be an expert on the game or a quick learner.

The table displayed in Fig. 6 summarizes all the developed difficulty modes and their strategies.

#### 4.2. Difficulty evaluation process

A difficulty evaluation process was developed to be performed during the game and indicates when the players are not evolving at the same pace. For that, it was necessary to observe which game features should be analyzed and how to properly use the information from each one of them. The analyzed features and the evaluation process are described below.

##### 4.2.1. Game features

To evaluate a game match, it is crucial to identify which features can represent the players' performance and are relevant to the evaluation. In our testbed, we identified three important features that can illustrate the player's behavior during a DotA match. These features are: Hero's Level, Hero's Death and Towers Destroyed. Each one of these features will be described in the following paragraphs:

*Hero's level.* This feature represents the player's evolution during a match, where the greater is the level value, the stronger is the character. Although this feature represents the evolution, it should not be the only analyzed feature because it is possible that the player increases his/her hero's level without really increasing his/her abilities. For example, the player can keep the hero closer to battles without engaging in any fight and, by doing that, it will gain some experience points that are shared among the allies that are closer to the battle and will help the hero to evolve its level. Thereby, even if all players have heroes with equivalent levels, this feature alone does not give a real track on the game balance.

*Hero's death.* This feature counts how many times the hero has died during a DotA match. Differently from all other features, the hero's death may represent the player's performance and the level of difficulty that he/she is facing more accurately. For example, an inexperienced player, even having a hero with a high level, may

have a high death rate, since he/she may not know how to use more properly the characteristics and peculiarities of his/her character as well as a possible lack of game strategies. Thereby, this feature seems to represent more accurately how well the player is facing the game challenges.

*Towers destroyed.* This feature is the amount of enemy's towers destroyed by the allied team. It represents the team expansion and dominance over the map. Although this feature is not directly related to the player's performance, since other allies can also destroy towers, it gives us a good notion of the game's progress and team expansion over the map. Therefore, if a team is quickly progressing over the map, it may represent that the game is unbalanced.

##### 4.2.2. Tracking player's performance

In order to perform a dynamic difficulty adjustment, it is necessary to evaluate the game from time to time and verify if the game is presenting challenges suitable to the player's performance. If the player is having a poor performance, the game should be capable to identify that and reduce its difficulty. In the same way, if the player evolves faster than the challenges presented, the game should increase its difficulty.

Once we have defined the game features that must be analyzed, this process can be summarized into the creation of an heuristic function that will keep track on the player's performance and inform when it is necessary to adjust the difficulty. This heuristic function will be our evaluation method during the game match and from now on it will be called as evaluation function. So, considering the features mentioned before and the impact that each one represents on the player's performance, we have:

$$P(x_t) = H_l - H_d + T_d, \quad (1)$$

where  $P(x_t)$  is the performance function of player  $x$  on time  $t$ .  $H_l$  is the hero's level,  $H_d$  is the hero's death count and  $T_d$  is the number of towers destroyed. It is important to mention that the values of these features are related to the player and his/her hero. Computing the difference between the measurements at two consecutive times  $t$  and  $t - 1$ , it is possible to calculate the current evolution of the player, as shown in the equation below:

$$P'(x) = P(x_t) - P(x_{t-1}). \quad (2)$$

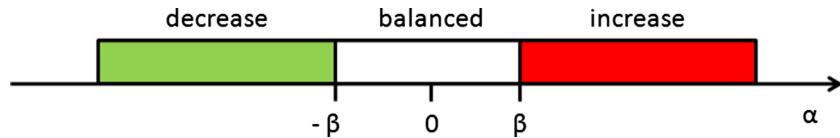
Once the performance function was calculated for both players ( $x$  and  $y$ ) the evaluation value can be obtained by:

$$\alpha = P'(x) - P'(y), \quad (3)$$

where  $\alpha$  is the difference between performances. We should mention that player  $x$  is the one that we are analyzing and player  $y$  is the one controlled by the artificial intelligence system. Therefore, the player  $y$  is the one that will have its difficulty adjusted during

|                  |                      | Artificial Intelligence |              |           |
|------------------|----------------------|-------------------------|--------------|-----------|
|                  |                      | Easy Mode               | Regular Mode | Hard Mode |
| Defense Strategy | Defend Allied Towers | X                       | X            | X         |
|                  | Retreat              | X                       | X            | X         |
|                  | Item Manipulation    |                         | X            | X         |
| Attack Strategy  | Main Attack          | X                       | X            | X         |
|                  | Target Selection     | X                       | X            | X         |
|                  | Track Enemy Hero     |                         |              | X         |
|                  | Cast Spell           |                         |              | X         |

Fig. 6. Summary of the developed strategies for each artificial intelligence agent.



**Fig. 7.** The verification performed by the adjustment mechanism.

the game. It is important to measure the opponent's performance in order to evaluate if its progress is compatible or not with the players.

#### 4.3. Dynamic difficulty adjustment mechanism

The proposed mechanism is the key to make the adjustment work properly during the game. Until now we have only showed how to verify if the player's performance is balanced to a certain opponent or not. Thereby, the main task of the implemented mechanism is to analyze the  $\alpha$  value and perform or not the difficulty adjustment at the game time  $t$ .

The mechanism works by evaluating the  $\alpha$  variable and constantly verifying if this variable is within the  $\beta$ 's range, where  $\beta$  represents the limit value of the evaluation function. This value means how far a player can perform better than the other player, without considering the game unbalanced. If the value of  $|\beta|$  is a large number, then the adjustment will occur with less frequency, since it may take some time to  $\alpha$  overcome  $\beta$ . Likewise, if  $|\beta|$  is a small number, then the adjustment will occur more frequently, since it may overcome  $\beta$  more easily. And if  $\alpha$  stays inside the limits values of  $-\beta$  and  $\beta$ , it means that both players are having a similar performance and therefore, the match is currently balanced. Fig. 7 illustrates this approach. In Section 5, several experiments were made in order to find the best limit value for  $\beta$ .

### 5. Experiments: agents vs agents

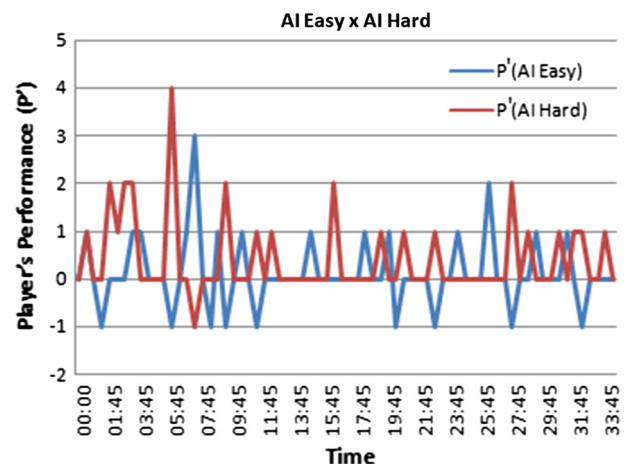
In order to verify the effectiveness of the proposed mechanism, a series of experiments was performed. The players' performance were analyzed along with the behavior of their heroes. The dynamic adjustment mechanism was also observed, as well as its variations and the impact caused on the matches.

On each experiment, we performed 20 matches of the game with the static artificial intelligence agent controlling one team against the dynamic artificial intelligence agent controlling the other one (Fig. 16, in the end of this section, shows a summary of the matches). The  $\beta$  limits for triggering the artificial intelligence switch were set to  $-1$  and  $1$ . Therefore, every time the difference among performances ( $\alpha$ ) exceeds the  $\beta$  limits, the difficulty of the dynamic AI should be modified accordingly. These values were determined empirically, after executing a large number of tests and observing the results contained in the gamelogs, and were not changed during these experiments.

#### 5.1. Baseline

First, we performed an unbalanced match in order to stipulate a baseline to compare with the obtained results from all three experiments. This baseline match is set by two different AI agent players with static behavior. One of them is on easy mode, representing a player without experience, and the second one is on hard mode, representing a very experienced player. The results of this match are shown in Fig. 8, where the difference among both performances can be noticed.

The player/agent performance is measured taking into account its current state during the match. The positive peaks represents



**Fig. 8.** Baseline values obtained from a match with static difficulty for both players.

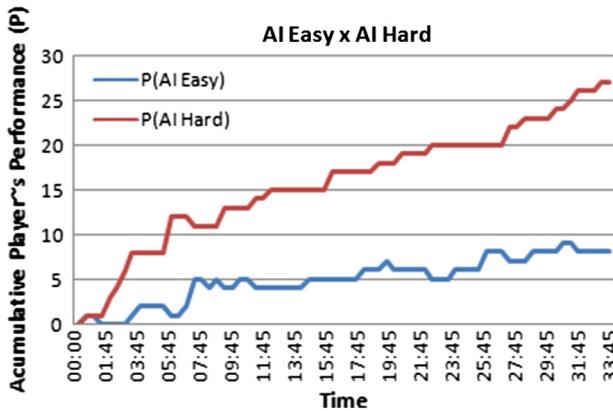
moments where the agent improved its performance when compared to its last state. Likewise, negative peaks mean that the agent had its performance decreased based on its last game state. Converting these to game situations, when a hero gains a level or the team manage to destroy a tower, then this will impact positively in its development, increasing the player's current performance. Similarly, if a hero dies this will result in a negative impact in its development decreasing the player's current performance.

During this game match, the hard mode player kept increasing his performance, presenting only one time of regression in his development. Meanwhile, the easy mode player performance was very unstable, with several moments of regression in his development. Therefore, we can consider that a match will be balanced if the difference among both performances were not divergent. So, examining once again the graph of Fig. 8, it is possible to observe that each performance peak shows itself as an appropriate moment to execute a difficulty balance in order to get the players' performance closer to each other.

Fig. 9 shows the cumulative performance value for each player during this particular match. On this graph, it becomes clear that the hard mode player evolves much faster than the easy mode player. This greater performance evolution can be related to the fact that the hero increases his level rapidly and has a low amount of deaths. On the other hand, the easy mode player had his performance lowered by a large number of deaths, in spite of having a good development. This led to a poor performance when compared to the hard mode player.

Therefore, due to that difference between them, the adjustment appears to be necessary in order to minimize this disparity among their behaviors and present a more fair and competitive game.

After setting a baseline of an unbalanced match, we performed a set of experiments to compare the performance of a player using the adaptive AI against an opponent using a static AI, fixed at one of the predetermined modes. Specifically, for player A we used a static artificial intelligence agent in order to simulate the possible skills of a human player. For player B we applied the proposed mechanism, so that this player should keep its performance equivalent to player A and for that it should perform a dynamic difficulty



**Fig. 9.** Cumulative player's performance with static difficulty for both players.

adjustment. These experiments are mentioned on the following subsections.

### 5.2. Easy × adaptive

In this first set of experiments, player A was set with the easy mode, simulating a novice player, while player B was set with the adaptive AI, starting with the regular mode and switching to try to match the other player performance. Fig. 10 shows the performance of both players during this match ( $P'$ ), while Fig. 11 shows the results of the evaluation function ( $\alpha$ ) and the difficulty adjustments made during the game.

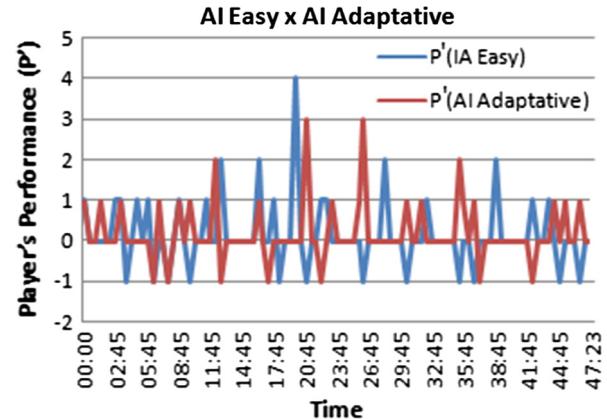
As mentioned before, the player's performance is measured by taking into account his/her current state during the match. The positive peaks represent moments where the player had improved and negative peaks mean that the player had decreased based on his/her last game state. Fig. 10 shows that the adaptive artificial intelligence agent (player B) managed to keep its performance similar to its opponent, the easy mode player A.

On Fig. 11, we can track how well the adaptive player (player B) managed to be compatible with player A during the match. When the evaluation function shows negative peaks, it means that the difficulty should be adjusted and decreased by one level. Likewise, if there are positive peaks, the difficulty of the adaptive player should be increased by one level. Moments where the evaluation function remains constant (equals 0) means that the performance of both players are very similar and due to that no adjustment is necessary at this time. Therefore, the difficulty can be maintained.

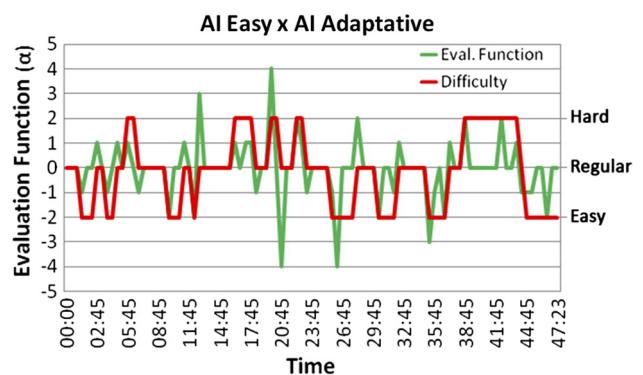
It is important to mention that the difficulty adjustment is performed by increasing or decreasing one level of each time. With this approach we minimize the possibility of the opponent player noticing the behavior change. After analyzing this set of experiments and studying the gamelogs obtained from each one, we observed that in 85% of the matches, the adaptive player B managed to keep the game balanced and as result of each match, player A won 60% of the matches and player B won 40%.

### 5.3. Regular × adaptive

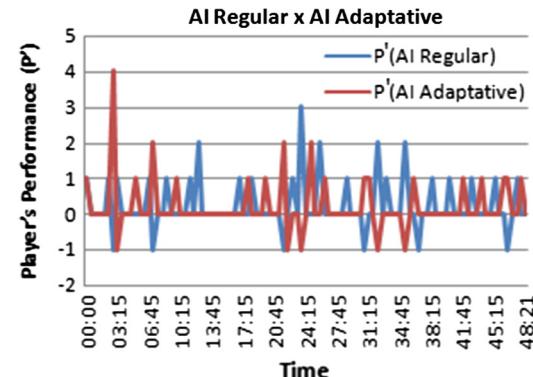
On the second set of experiments, we kept using the artificial intelligence agents developed to control two players, one from each team. Here, we manage to simulate an intermediary player with player A using a static artificial intelligence agent on regular mode. For player B we applied the proposed mechanism, starting it on regular mode. Fig. 12 shows the performance of both players ( $P'$ ) during one single match. Likewise, Fig. 13 shows the results of the evaluation function ( $\alpha$ ) during the game and the difficulty adjustments made along the match.



**Fig. 10.** Performance of easy × adaptive players during one match.



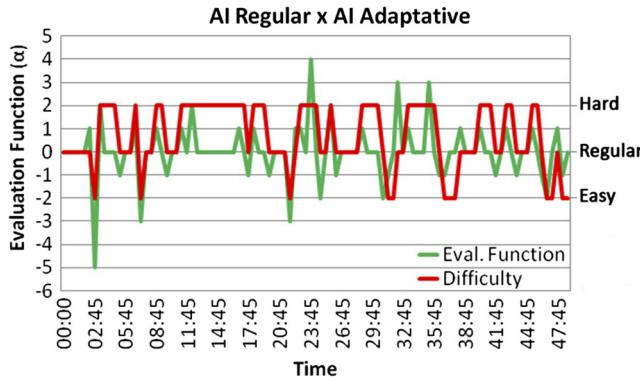
**Fig. 11.** Difficulty adjustments performed by the mechanism during one match.



**Fig. 12.** Performance of regular × adaptive players during one match.

The analysis performed in this set of experiments is pretty similar to the previous one. The positive peaks represent moments where the player had improved and negative peaks mean that the player had decreased its performance. In Fig. 12, we can observe that the adaptive artificial intelligence agent (player B) tried to follow its opponent's performance (player A) presenting similar peaks at close time periods.

On Fig. 13, we can follow all the adjustments made during the match. The adaptive player spent most of its time alternating between the regular mode and the hard mode. This variation can be understood as moments where player B was having a poor development when compared to player A, and the need to increase the difficulty was perceived. Similarly, when player's B behavior were standing out, the need for reducing the difficulty could also be seen. The graphic also shows that player B stayed balanced



**Fig. 13.** Difficulty adjustments performed by the mechanism during one match.

during the game. Furthermore, after analyzing this second set of experiments and studying all gamelogs collected, we observed that the players had a compatible performance in 90% of the matches. The results of the matches were 50% of victories for each player.

#### 5.4. Hard × adaptive

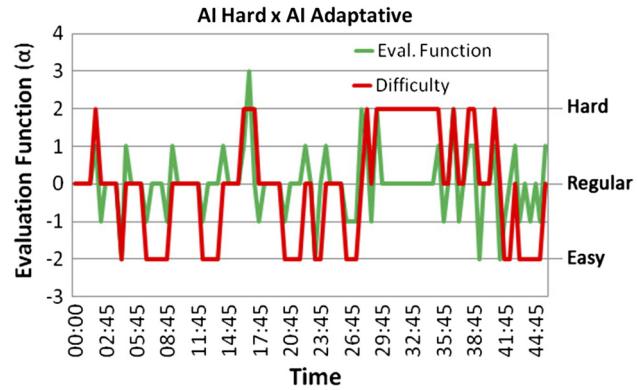
On the last set of experiments, we managed to simulate an expert player (player A) against the adaptive player (player B). As we mentioned before, the adaptive player started on regular mode and changed its behavior during the match in order to keep the game balanced. Fig. 14 shows the performance ( $P'$ ) of both players during one match. Likewise, Fig. 15 shows the results of the evaluation function ( $\alpha$ ) during the game and the difficulty adjustments made along the match.

Analyzing the results from Fig. 14, the adaptive player started developing a better performance than player A in the beginning of the match. Therefore, it was detected that the difficulty should be reduced in order to keep the balance (Fig. 15). After that, they kept their performances very close and the difficulty kept alternating between easy mode and regular mode until player A can present itself better/stronger than player B. The opposite can also be seen, when player B kept alternating between regular mode and hard mode in order to reach player's A performance.

Furthermore, after analyzing the gamelogs collected from each game, we observed that the adaptive player (player B) changed its difficulty and succeed to keep the match balanced on 80% of the experiments. As result of the battles, player A won 45% of the matches.

#### 5.5. Discussion

Despite the good results, not all the cases presented the expected results, which has resulted in unbalanced matches. To



**Fig. 15.** Difficulty adjustments performed by the mechanism during one match.

| AI Adaptive       |     |      |          |
|-------------------|-----|------|----------|
|                   | Win | Lose | Win rate |
| <b>AI Easy</b>    | 8   | 12   | 40%      |
| <b>AI Regular</b> | 10  | 10   | 50%      |
| <b>AI Hard</b>    | 11  | 9    | 55%      |

**Fig. 16.** Final results from all matches performed on the experiments.

get to this conclusion, we observed all the executed matches and studied all the collected gamelogs. These gamelogs kept track of the game on every 15 s, recording the current situation of both teams, the related features, their values, among other information. Once the game was finished, we started to translate those collected information, comparing the values from both heroes and making the necessary assumptions.

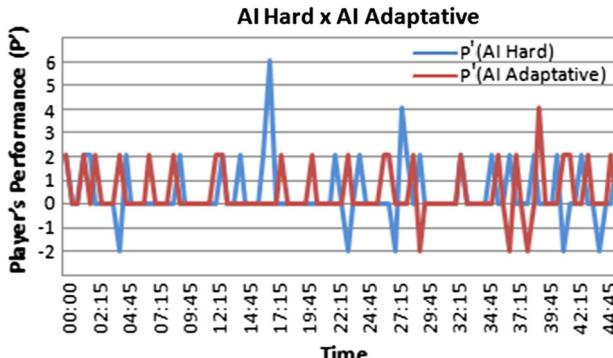
Considering all the performed experiments 10% of them were unbalanced because the mechanism took too long to perform each adjustment, leading to a great difference between the players performance. So, when the players were getting closer to a balance, the match has ended. On the other hand, 5% of the executed experiments were unbalanced due to an excess of adjustments. In these scenarios, the adjustments were being performed too quickly, causing player B to not evolve properly during the match, which resulted in an easy game for player A.

After performing all the experiments, it was possible to summarize the obtained final results from the game matches. Fig. 16 shows the amount of victories and losses of the adaptive AI against the easy, regular and hard modes. These results show that both players had a similar amount of victories, demonstrating that the adaptive player offers a more balanced game experience.

## 6. Experiments: agents vs users

We also performed tests with users to assess qualitatively the efficacy of the implemented mechanism. The objective of these tests consists of verifying, according to the perception of the player, if the mechanism can keep the difficulty of the game balanced facing their skills and if this approach stimulates his/her entertainment. To make the tests more objective, we only enrolled users who have played the game DEFENSE OF THE ANCIENTS (DotA) at least once, in order to avoid problems in understanding interface elements and/or the game's mechanics.

The tests involved inviting the users to play two matches of the game DEFENSE OF THE ANCIENTS (DotA), where their main goal was to defeat the opposing team. However, it was explained to them that



**Fig. 14.** Performance of hard × adaptive players during one match.

the outcome of the match was not crucial for the experiment, since we were more interested in behavioral issues generated by the game. We provided two types of maps. On map A, users faced the dynamic agent as opponent and on map B they faced a static agent. With these two types of maps, we evaluated if the difficulty adjustment can be perceived during the match and whether or not this dynamic adjustment can really impact on the player entertainment. Note that each volunteer played two times on the same map, not changing maps. We performed the tests on this way because we were trying to avoid the placebo effect on the users. Once the user knows that he was playing in both game versions he could assume that the adjustment was happening even though it may not be true. Therefore, we managed to ask only the expert users to play against both opponents (dynamic agent and static agent), in order to really assess if the adjustment can be detected or not. We choose to ask only the expert players to play in both maps because they have more knowledge and expertise in the game mechanics, this makes them able to identify quickly any abnormal behavior on the opponent.

A total of eleven users participated on the tests. All tests were guided by the following script:

1. The participant was instructed about how the test was going to occur and signed an agreement to participate on the test.
2. A quick presentation of the game mechanics was carried out and its rules in case the user did not remember it.
3. A preliminary interview was made, whose main objective was to let us know the user's profile and background regarding game playing.
4. The participant played two game matches of DotA on the same map.
5. A post-test interview was made, whose objective was to gather the user's opinions and perceptions regarding the presented system.

The results from these tests are discussed in the following sections.

### 6.1. Pre-test evaluation

During the tests, all users were instructed to answer a series of questions to make it possible to establish profiles according to their habits, preferences and experiences. These questions were placed on the pre-test questionnaire and its main goal is to try to

identify similarities among the volunteers' behavior during the experiments.

A total of eleven male users participated on the experiments. Among them one is less than 18 years old, two are from 18 to 21 years old, four are from 22 to 25 years old and the last four are between 26 and 29 years old. Regarding their education level, one was still in high school, seven were studying or already concluded a higher education and one had master's degree.

To verify their experience with digital games, we asked which are the game genres that the users have recently played the most. The most popular genres were platform games and RPG selected by 90.9% of the volunteers, followed by First Person Shooter (FPS) and sports with 81.8% of the users and on third place there is Real-Time Strategy (RTS) and racing with 72.2%. Observe that this percentages do not sum up, because users could select more than one option. Fig. 17 shows all the genres selected by the volunteers.

Regarding the frequency on which they play, 63.6% of the users play every day and 36.4% from 1 to 3 times per week. The players were also asked on which devices they are currently playing more and 90.9% of the users selected mobile devices, followed by PC with 81.8% and consoles with 36.4%. Fig. 18 presents the chart with the most popular devices.

Now, regarding MOBA games knowledge, all users said that they had played DotA at least once, on which five rated themselves as beginners, four as intermediary players and two as experts. Among the eleven volunteers, only eight said they had played another MOBA game: 100% of them played *League of Legends* and 37.5% played *Heroes of Newerth*. About their expertise in MOBA games in general, 25% self-proclaimed as beginners, 37.5% as intermediary players and 37.5% as expert players.

### 6.2. Post-test evaluation

After answering all pre-test questions, the participants were asked to play two matches against our agents. Therefore, we created two different maps (A and B) and asked the volunteer to play both matches in only one map. Both maps contain the same game structure and what differ them is that map A hosts the dynamic artificial intelligence agent while map B hosts the static artificial intelligence agent. These agents are the same used on the previous experiment. Among the participants, six of them played on map A and five of them played on map B.

Once both matches were concluded, the user was instructed to answer the post-test questionnaire, which tackles various aspects

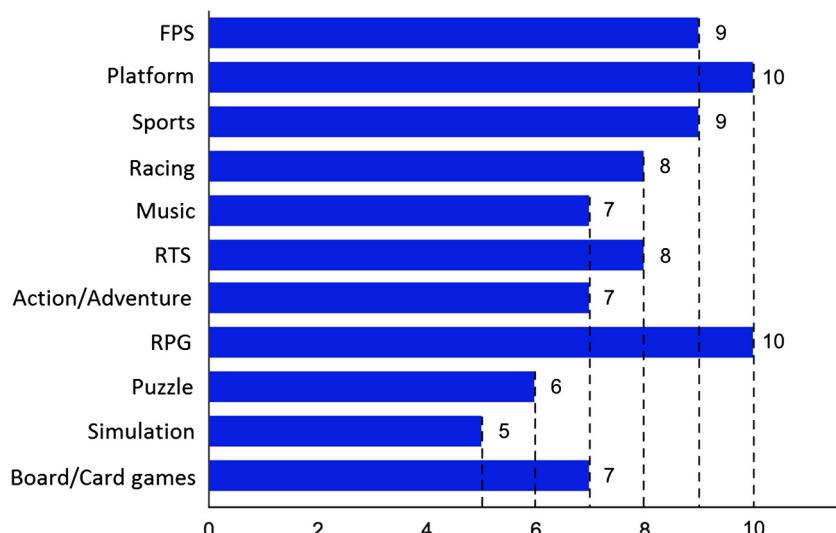
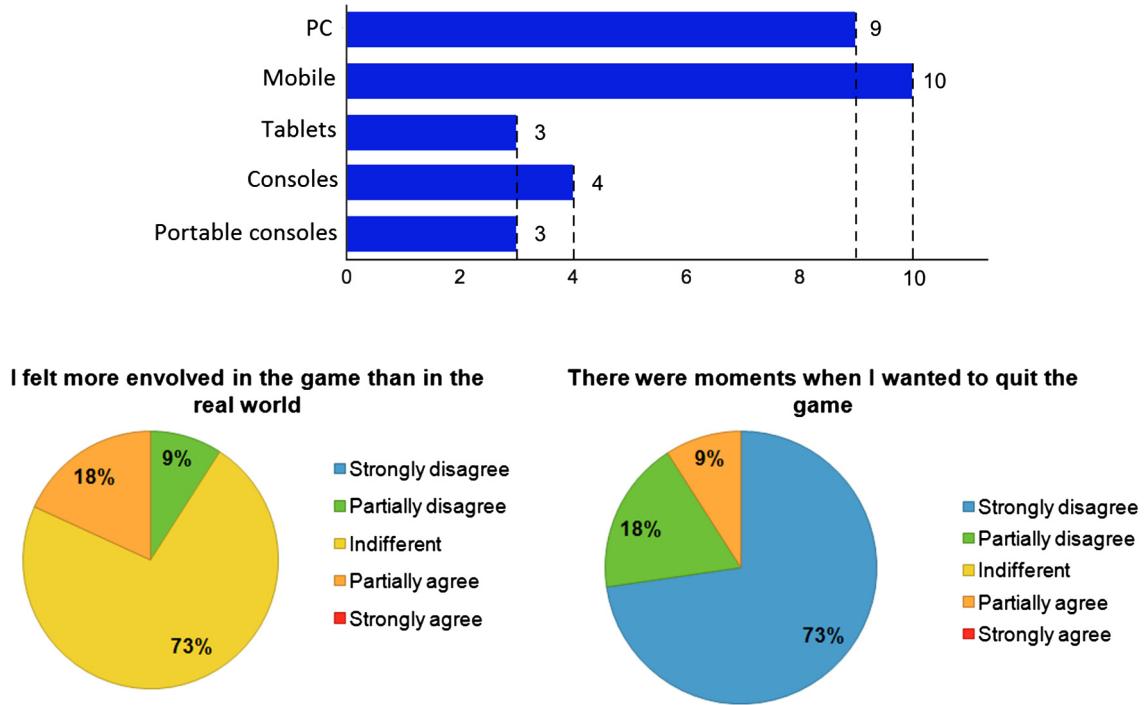


Fig. 17. Most popular game genres among the volunteers.



**Fig. 19.** Affirmatives that address aspects of the player immersion during the match.

related to how he/she perceived the game experience. Questions related to experience and immersion of the user during the match came from a selection of questionnaires about user experience in games [40,14,19,18]. They were presented as affirmatives so that the user could choose how much he/she agrees to it following the 5 points of *Likert's* classification [31]. The scale goes from 1 - "Strongly disagree"; 2 - "Partially disagree"; 3 - "Indifferent"; 4 - "Partially agree"; and 5 - "Strongly agree".

The first set of affirmatives addresses aspects of the player immersion during the match: 55% described as indifferent when asked if they did not realize the time running while playing and 73% of them strongly agree that they worked hard to get good results in the game. When asked if there were moments where they wanted to quit the game, 73% strongly disagree with this affirmative. This may suggest that although half of the volunteers said they were indifferent regarding the game time lapse, the majority stated that they put some effort to accomplish the main goal and they did not want to quit (Fig. 19). Thereby, we can assume that the game was able to offer a considerable level of immersion to the player.

The next set of affirmatives addresses the game challenge provided by the agent during the match. When asked if the game kept them motivated to keep playing, 18% strongly agree and 46% partially agree. Now, regarding the difficulty, 27% strongly agree that the game is too challenging for them, while 18% strongly agree that the game is suitably challenging for them. Finally, 36% said that the game is not challenging at all (Fig. 20). These very distinct opinions can be observed due to the different level of expertise of each player. Those that consider themselves as beginners in DotA or general MOBA, found that the game was too challenging for them. Using the same analysis, those that consider themselves as experts found the game too easy because they know more advanced strategies that goes beyond the agent's algorithm.

The following set of affirmatives addressed the player ability/competence during the game. 64% of the players felt successful at the end of the game and won the match. Also, 82% agree that they were making progress during the course of the game. About the player enjoyment during the game, 73% liked to play against

our agent and would recommend this game to others. 64% said that he would play this game again (Fig. 21). Therefore, although the agents were not always very challenging to all players, we can assume that most of them enjoyed the match against it.

The last set of affirmatives addresses the agent opponent level. Here, all the opinions got divided where we have that 27.3% strongly agree and 9.1% partially agree that the opponent plays a lot better than they. This opinion were given probably by beginners, since they may had a more difficult time trying to win the match. Regarding that the opponent plays in the same level as them, 27.3% strongly agree with this affirmation, which may indicate that these players are intermediaries or maybe beginners that managed to win the match. Finally, we have that 36.4% of the players strongly agree that the opponent plays much worse than them. This answer was given by experts and intermediaries players that managed to win the match without putting too much effort on this accomplishment. Thereby, we can observe that the player opinion about the opponent reflects directly on his/her expertise, where the agent with the same behavior can be too challenging for some and not very challenging for another.

To analyze the player's perception during the game, we asked them some questions about whether or not they have noticed any changes on the agent's behavior during the match. We made these questions to all the players regardless the map where they played. All of them said they did not notice the opponent getting harder or easier. They also mentioned not noticing the agent adapting its behavior in order to be more suitable to them. Therefore, based on these answers we can assume that the mechanism were successful in at least one of its goals by changing its behavior without making it noticeable to the player. This is the first premise of the dynamic difficulty adjustment which tries to guarantee that the player will not feel cheated or disappointed during the game.

### 6.3. Discussion

The main goal of these experiments were to evaluate qualitatively the efficacy of the implemented mechanism, verifying if it can keep the game difficulty balanced to each player, and whether

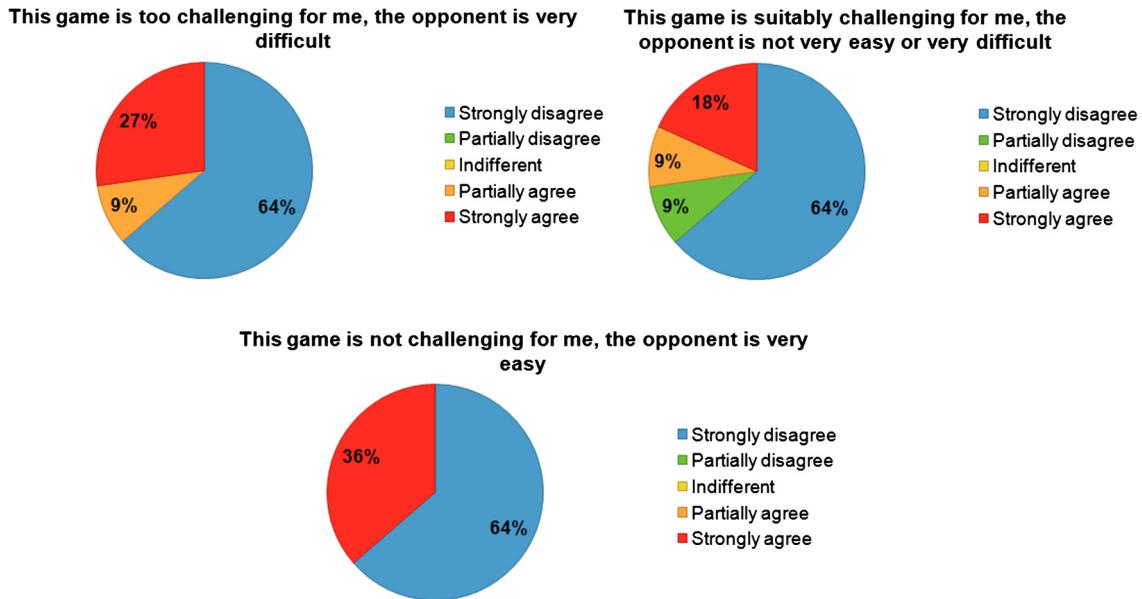


Fig. 20. Affirmatives that address the game challenge provided by the agent during the match.

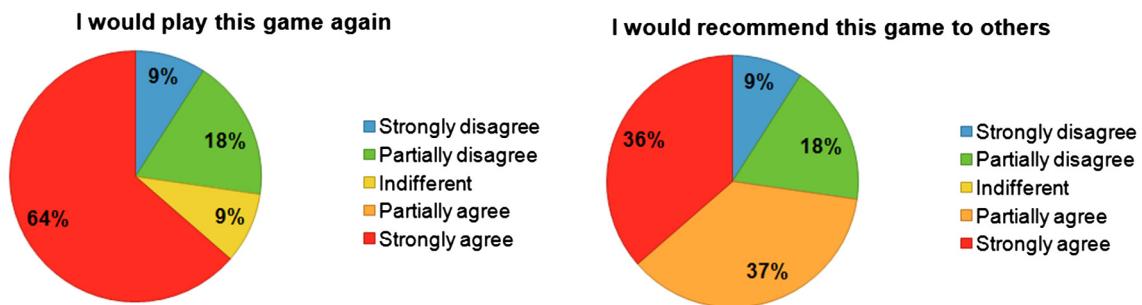


Fig. 21. Affirmatives that address the enjoyment during the match.

or not this can really impact on the player entertainment. Therefore, we provided two types of maps, one with the dynamic agent and another with the static agent.

After analyzing the results from all players, we asked informally for both expert volunteers to also play on the other map, so that we could assess their perceptions regarding both agents. After comparing the experience they had on each map, they said that the opponent from map A (dynamic agent) presented a more fluid behavior and they preferred to play against it. They also stated that both maps were not very challenging to them and maybe it could be more suitable for players with little experience. With that feedback we believe that, in the current state, our agent has enough expertise to play against novice players. Thereby, the novice players can be entertained while learning the game style.

By observing the data regarding the player immersion during the match, we can consider that the game provided a satisfactory level of immersion, since none of the players said they wanted to quit the game or found it too long. As well as the questions related to the player enjoyment during the match, although the agents were not always very challenging against all users, we can assume that most of them enjoyed playing against it.

However, after considering the responses related to the game challenge provided, it was observed that, for experts, the developed artificial intelligence agent turned out to be weak and not very challenging. We believed that this occurred due to the absence of more complex strategies during its development, once

it is not a simple task to develop such agents. As mentioned on Section 3.6, developing agents capable of defeating competitive human players in MOBA remains an open research challenge and we can attribute our agents flaws to this. Some of the intermediary players also found both agents not very difficult to defeat. As for the entry-level players, based on their responses we could state that they found both opponents too challenging, because probably they had a more difficult time trying to win the match although the agent had kept the same pace as his.

## 7. Conclusion

The dynamic difficulty adjustment consists in an alternative towards the definition of the game challenge levels. This adjustment is dynamically performed, making it possible to track the player's skills and adjust itself during game runtime.

The presented work aimed to increase the player's entertainment by providing a mechanism that adjusts the game AI agent according to the player's skills. This mechanism was implemented on a MOBA game, called DEFENSE OF THE ANCIENT (DotA). After performing experiments that simulate the three main player's behaviors (beginner, regular and experienced), it was possible to verify that the dynamic difficulty adjustment mechanism was able to keep up with the player's abilities on 85% of all experiments. On the remaining experiments that failed to suit the player's skill, 10%

of it occurred because the adjustment mechanism spent too much time to perform each needed adjustment which resulted in a great difference between the players performance. And the last 5% of it occurred due to an excess of adjustments that were performed too quickly, without giving enough time to the game to evolve properly.

Given the presented results, we can conclude that the proposed mechanism behaved as expected and is capable to offer a game match compatible with the simulated player's performance. Also, after observing all obtained results, we can state that the key to a balanced game is to keep changing the difficulty of the adaptive player in order to follow the performance of the human player and avoid boredom and frustration.

As future work, the dynamic difficulty adjustment mechanism will be improved in order to decrease the amount of cases where the balance did not work properly. We believe that we must implement a general AI, capable of handling almost all heroes available in DotA, in order to offer diversity of strategies and helping the player to learn how to play against different heroes. Another improvement that could be done is testing the activation or deactivation of features, like item buy, spell casting and combos, tracking the player behavior and trying to imitate his/her knowledge.

In machine learning field, it is possible to try to learn the players preference, in order to improve the knowledge of the intelligent agent, making it to follow the player's behavior. Moreover, machine learning could track which skills the player has developed and push new knowledge into the agent in order to try to stimulate the player to explore the game and learn what the agent is doing. We could also track the player preferences in order to try to classify his/her play style and select heroes that match his/her preferences, making the learning curve smoother. Finally we want to perform qualitative tests with a larger number of players in order to get more insights about the proposed mechanism.

### Acknowledgements

This work was partially supported by CAPES, CNPq and FAPE-MIG. We would like to thank all the volunteers that participated in our qualitative tests.

### Appendix A. User experiments questionnaire

See Figs. A.22,A.23,A.24,A.25,A.26,A.27,A.28,A.29.

### Questionnaire Pre Test

This questionnaire must be completed BEFORE you start playing DotA game.

**Player Gender \***

♂    ♀

**Age \***

Less than 18 years old  
 18 to 21 years old  
 22 to 25 years old  
 26 to 29 years old  
 More than 30 years old

**What is your educational level? (Complete or Incomplete) \***

High School  
 Higher education  
 Postgraduate studies

**What types of games you have played? (If necessary, choose more than one) \***

FPS (Counter Strike, Doom, Call of Duty, Medal of Honor, etc.)  
 Platform (Sonic, Mario Bros, Metroid, etc.)  
 Sports  
 Racing  
 Music (Guitar Hero, Rock Band, etc.)  
 RTS (Age of Empires, Warcraft, Starcraft, etc.)  
 Action/Adventure (Assassin's Creed, Tomb Raider, Uncharted, etc)  
 RPG (Zelda, Final Fantasy, Ragnarök, Diablo, etc.)  
 Puzzle (Tetris, Angry Birds, etc.)  
 Simulation (Sim City, The Sims, etc.)  
 Cards and Board (Chess, Solitaire, etc.)  
 Outro:

**Fig. A.22.** Pre-test questionnaire.

**Which platform do you currently play? (If necessary, choose more than one) \***

Computer  
 Mobile  
 Tablets  
 Consoles (PS4, Xbox One, Wii U, etc.)  
 Portable consoles (Game Boy, PSP, Nintendo DS, etc.)

**How often do you play? \***

Everyday  
 1 to 3 times per week  
 1 to 3 times per month

**Have you ever played Defense of the Ancient (DotA)? \***

**How would you rate yourself as a DotA player? \***

Beginner  
 Intermediary  
 Expert

**You have played another MOBA game? \***

Yes  
 No

**If you already played another MOBA genre, enter the name of it below:**

**How would you rate yourself as a MOBA player? \***

Beginner  
 Intermediary  
 Expert

**Fig. A.23.** Pre-test questionnaire.

## Post-test questionnaire

This questionnaire must be completed AFTER you finish every match of DotA game.

**Which map you played? \***

### Game Immersion

**I did not realize the time running while playing. \***

1 2 3 4 5

Strongly disagree      Strongly agree

**I felt more involved in the game than in the real world. \***

1 2 3 4 5

Strongly disagree      Strongly agree

**I worked hard to get good results in the game \***

1 2 3 4 5

Strongly disagree      Strongly agree

**There were moments when I wanted to quit the game. \***

1 2 3 4 5

Strongly disagree      Strongly agree

**The game took too long to end. \***

1 2 3 4 5

Strongly disagree      Strongly agree

**Fig. A.24.** Post-test questionnaire: questions about immersion.

## Game Challenge

**During the match, I felt anxious. \***

1 2 3 4 5

Strongly disagree      Strongly agree

**During the match, I felt bored. \***

1 2 3 4 5

Strongly disagree      Strongly agree

**The game kept me motivated to keep playing. \***

1 2 3 4 5

Strongly disagree      Strongly agree

**My skills have improved gradually by overcoming the challenges. \***

1 2 3 4 5

Strongly disagree      Strongly agree

**The game offers new challenges in an appropriate pace. \***

1 2 3 4 5

Strongly disagree      Strongly agree

**This game is too challenging for me, the opponent is very difficult. \***

1 2 3 4 5

Strongly disagree      Strongly agree

**Fig. A.25.** Post-test questionnaire: questions about challenge.

**This game is suitably challenging for me, the opponent is not very easy or very difficult. \***

1 2 3 4 5

---

Strongly disagree      Strongly agree

**This game is not challenging for me, the opponent is very easy. \***

1 2 3 4 5

---

Strongly disagree      Strongly agree

**Player Skills/Habiliites**

**I felt successful. \***

1 2 3 4 5

---

Strongly disagree      Strongly agree

**I reached the goal of the game. \***

1 2 3 4 5

---

Strongly disagree      Strongly agree

**I felt competent. \***

1 2 3 4 5

---

Strongly disagree      Strongly agree

**I felt that I was making progress during the course of the game. \***

1 2 3 4 5

---

Strongly disagree      Strongly agree

**Fig. A.26.** Post-test questionnaire: questions about player skills.

## Enjoyment

**I liked to play against this opponent. \***

1 2 3 4 5

Strongly disagree      Strongly agree

---

**When stopped, I was disappointed that the game was over. \***

1 2 3 4 5

Strongly disagree      Strongly agree

---

**I would play this game again. \***

1 2 3 4 5

Strongly disagree      Strongly agree

---

**I would recommend this game to others. \***

1 2 3 4 5

Strongly disagree      Strongly agree

---

**Some things in the game annoyed me. \***

1 2 3 4 5

Strongly disagree      Strongly agree

---

**It was a boring match. \***

1 2 3 4 5

Strongly disagree      Strongly agree

**Fig. A.27.** Post-test questionnaire: questions about enjoyment during the match.

## Opponent Level

**The opponent plays a lot better than me. \***

1 2 3 4 5

Strongly disagree      Strongly agree

---

**The opponent plays the same proportion as me. \***

1 2 3 4 5

Strongly disagree      Strongly agree

---

**The opponent plays much worse than me. \***

1 2 3 4 5

Strongly disagree      Strongly agree

**Fig. A.28.** Post-test questionnaire: questions about the opponent level.

**Game Perceptions**

**Have you noticed any change in the opponent's behavior so that it has adapted to suit your performance? \***  
 (YES or NO - If you answer YES, what was your perception?)

**Did you notice the opponent getting easier during the game? \***  
 (YES or NO - If you answer YES, what was your perception?)

**Did you notice the opponent getting harder during the game? \***  
 (YES or NO - If you answer YES, what was your perception?)

**Leave your comments here:**

**Fig. A.29.** Post-test questionnaire: questions about the player's perception during the game match.

## References

- [1] G. Andrade, G. Ramalho, H. Santana, V. Corruble, Extending reinforcement learning to provide dynamic game balancing, in: Proceedings of the Workshop on Reasoning, Representation, and Learning in Computer Games, 19th International Joint Conference on Artificial Intelligence (IJCAI), 2005, pp. 7–12.
- [2] S. Bakkes, P. Spronck, J. van den Herik, Rapid and reliable adaptation of video game AI, *IEEE Trans. Comput. Intell. AI Games* 1 (2) (2009) 93–104.
- [3] R.A. Bartle, Designing Virtual Worlds, New Riders, 2004.
- [4] D.A. Bowman, R.P. McMahan, Virtual reality: how much immersion is enough?, *Computer* 40 (7) (2007) 36–43.
- [5] M. Buro, Real-time strategy games: a new AI research challenge, in: IJCAI, 2003, pp. 1534–1535.
- [6] M. Buro, Call for ai research in RTS games, in: Proceedings of the AAAI-04 Workshop on Challenges in Game AI, 2004, pp. 139–142.
- [7] M. Buro, T. Furtak, RTS games and real-time AI research, *Proceedings of the Behavior Representation in Modeling and Simulation Conference (BRIMS)*, vol. 6370, 2004.
- [8] M. Csikszentmihalyi, Flow, HarperCollins, 1991.
- [9] M. Csikszentmihalyi, Beyond Boredom and Anxiety, Jossey-Bass, 2000.
- [10] M. Csikszentmihalyi, J. Nakamura, Effortless attention in everyday life: a systematic phenomenology, in: B. Bruya (Ed.), *Effortless Attention: A New Perspective in the Cognitive Science of Attention and Action*, MIT Press, 2010, pp. 179–189.
- [11] B.B.P.L. de Araujo, B. Feijó, Evaluating dynamic difficulty adaptivity in shoot'em up games, in: *Proceedings of the XII Brazilian Symposium on Games and Digital Entertainment - SBGames 2013*, São Paulo, Brazil, 2013, pp. 229–238.
- [12] P. Demasi, J.d.O. Adriano, On-line coevolution for action games, *Int. J. Intell. Games Simul.* 2 (2) (2003).
- [13] S. Egenfeldt-Nielsen, J.H. Smith, S.P. Tosca, *Understanding Video Games: The Essential Introduction*, Routledge, 2013.
- [14] C.M. Fox, J.H. Brockmyer, The development of the game engagement questionnaire: a measure of engagement in video game playing: response to reviews, *Interact. Comput.* (2013) iwt003.
- [15] R. Games, W.R. Games, League of legends, *Comput. Game* (2009).
- [16] J. Gaudiosi, Riot games' league of legends officially becomes most played pc game in the world, *Forbes* July 11, 2011, 2012.
- [17] R. Hunicke, The case for dynamic difficulty adjustment in games, in: *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology*, ACM, 2005, pp. 429–433.
- [18] W. Ijsselsteijn, Y. De Kort, K. Poels, A. Jurgelionis, F. Bellotti, Characterising and measuring user experiences in digital games, *International Conference on Advances in Computer Entertainment Technology*, vol. 2, 2007, p. 27.
- [19] C. Jennett, A.L. Cox, P. Cairns, S. Dhoparee, A. Epps, T. Tijs, A. Walton, Measuring and defining the experience of immersion in games, *Int. J. Human-Comput. Stud.* 66 (9) (2008) 641–661.
- [20] D. Johnson, L.E. Nacke, P. Wyeth, All about that base: differing player experiences in video game genres and the unique case of moba games, in:

- Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, ACM, 2008, pp. 2265–2274.
- [21] C. Klimmt, C. Blake, D. Hefner, P. Vorderer, C. Roth, Player performance, satisfaction, and video game enjoyment, in: Entertainment Computing—ICEC 2009, Springer, 2009, pp. 1–12.
- [22] R. Koster, Theory of fun for game design, O'Reilly Media, Inc., 2010.
- [23] H. Kwak, J. Blackburn, S. Han, Exploring cyberbullying and other toxic behavior in team competition online games, *Soc. Dyn.* 22 (28) (2015) 47.
- [24] H.S. Levoll, J. Vittersø, Can balance be boring? A critique of the challenges should match skills hypotheses in flow theory, *Soc. Indicator. Res.* 115 (1) (2014) 117–136.
- [25] M.C. Machado, E.P. Fantini, L. Chaimowicz, Player modeling: towards a common taxonomy, in: 2011 16th International Conference on Computer Games (CGAMES), IEEE, 2011, pp. 50–57.
- [26] T.W. Malone, Toward a theory of intrinsically motivating instruction\*, *Cognit. Sci.* 5 (4) (1981) 333–369.
- [27] M. Mateas, Interactive Drama, Art and Artificial Intelligence, Carnegie Mellon University, Pittsburgh, PA, USA, 2002, aAI3121279.
- [28] O. Missura, T. Gärtner, Player modeling for intelligent difficulty adjustment, in: Discovery Science, Springer, 2009, pp. 197–211.
- [29] L. Nacke, C.A. Lindley, Flow and immersion in first-person shooters: measuring the player's gameplay experience, in: Proceedings of the 2008 Conference on Future Play: Research, Play, Share, ACM, 2008, pp. 81–88.
- [30] A. Nareyek, AI in computer games, *Queue* 1 (10) (2004) 58.
- [31] G. Norman, Likert scales, levels of measurement and the laws of statistics, *Adv. Health Sci. Educ.* 15 (5) (2010) 625–632.
- [32] S. Poole, Trigger Happy: Videogames and the Entertainment Revolution, Arcade Publishing, 2004.
- [33] M.P. Silva, Inteligência Artificial Adaptativa Para Ajuste dinâmico de dificuldade em jogos digitais, Master's thesis, Universidade Federal de Minas Gerais (UFMG), Belo Horizonte/MG, Brazil, 2015.
- [34] M.P. Silva, V.d.N. Silva, L. Chaimowicz, Dynamic difficulty adjustment through an adaptive AI, in: Brazilian Symposium on Games and Entertainment (SBGames), 2015 SBC 14th, 2015, pp. 52–59.
- [35] V.d.N. Silva, L. Chaimowicz, On the development of intelligent agents for moba games, in: Brazilian Symposium on Games and Entertainment (SBGames), 2015 SBC 14th, 2015.
- [36] V. d. N. Silva, L. Chaimowicz, A tutor agent for moba games, in: Brazilian Symposium on Games and Entertainment (SBGames), 2015 SBC 14th, 2015.
- [37] A.M. Smith, C. Lewis, K. Hullett, G. Smith, A. Sullivan, An Inclusive Taxonomy of Player Modeling, University of California, Santa Cruz, Tech. Rep. UCSC-SOE-11-13, 2011.
- [38] P. Spronck, M. Ponsen, I. Sprinkhuizen-Kuyper, E. Postma, Adaptive game ai with dynamic scripting, *Mach. Learn.* 63 (3) (2006) 217–248.
- [39] K.O. Stanley, B.D. Bryant, R. Miikkulainen, Evolving neural network agents in the nero video game, *Proc. IEEE* (2005) 182–189.
- [40] J. Takatalo, J. Häkkinen, G. Nyman, Understanding presence, involvement, and flow in digital games, in: Game User Experience Evaluation, Springer, 2015, pp. 87–111.
- [41] P. Thompson, R. Parker, S. Cox, Interrogating creative theory and creative work: inside the games studio, *Sociology* (2015), 0038038514565836.
- [42] J. Togelius, R. De Nardi, S.M. Lucas, Towards automatic personalised content creation for racing games, in: IEEE Symposium on Computational Intelligence and Games, 2007, CIG 2007, IEEE, 2007, pp. 252–259.
- [43] W. Van Den Hoogen, W. Ijsselsteijn, Y. de Kort, Exploring behavioral expressions of player experience in digital games, in: Proceedings of the Workshop on Facial and Bodily Expression for Control and Adaptation of Games ECAG 2008, 2008, pp. 11–19.
- [44] B.G. Weber, M. Mateas, A. Jhala, Applying goal-driven autonomy to starcraft, in: AIIDE, 2010.
- [45] B.G. Weber, M. Mateas, A. Jhala, Building human-level AI for real-time strategy games, AAAI Fall Symposium: Advances in Cognitive Systems, vol. 11, 2011, p. 01.
- [46] D. Wheat, M. Masek, C.P. Lam, P. Hingston, Dynamic difficulty adjustment in 2d platformers through agent-based procedural level generation, in: 2015 IEEE International Conference on Systems, Man, and Cybernetics (SMC), IEEE, 2015, pp. 2778–2785.
- [47] G. Xavier, A condição eletrolúdica: cultura visual nos jogos eletrônicos, Novas ideias, Teresópolis, 2010.
- [48] G.N. Yannakakis, How to model and augment player satisfaction: a review, in: Proceedings of the 1st Workshop on Child, Computer and Interaction (ICMI'08), ACM Press, Montreal, Canada, 2008.
- [49] G.N. Yannakakis, J. Hallam, Towards optimizing entertainment in computer games, *Appl. Artif. Intell.* 21 (10) (2007) 933–971.

# Optimization of Platform Game Levels for Player Experience

**Chris Pedersen, Julian Togelius, Georgios Yannakakis**

IT University of Copenhagen  
Rued Langgaards Vej 7, DK-2300 Copenhagen S  
Denmark  
gammabyte@gmail.com, {juto, yannakakis}@itu.dk

## Abstract

We demonstrate an approach to modelling the effects of certain parameters of platform game levels on the players' experience of the game. A version of Super Mario Bros has been adapted for generation of parameterized levels, and experiments are conducted over the web to collect data on the relationship between level design parameters and aspects of player experience. These relationships have been learned using preference learning of neural networks. The acquired models will form the basis for artificial evolution of game levels that elicit desired player emotions.

## Introduction

Numerous theories exist regarding what makes computer games **fun**, as well as which aspects contribute to other types of player experience (Csikszentmihalyi 1990; Koster 2005). Recently, research in player satisfaction modelling has focused on empirically measuring the effects on player experience of changing various aspects of computer games, such as NPC playing styles (Yannakakis and Hallam 2007). Such studies have been conducted using both in-game data collection, questionnaires and physiological measurements (Yannakakis and Hallam 2008a).

A parallel research direction aims to find methods for **automatically generating entertaining game content**. These efforts see some aspect of a game as variable, defines a **fitness** ("goodness") function based on a theory of player satisfaction, and uses a learning or **optimization algorithm** to change the variable aspect of the game so as to **become more "fun"** according to the definition. The few published papers on this topic deal with optimizing narrative (Nelson, Ashmore, and Mateas 2006), racing tracks (Togelius, De Nardi, and Lucas 2007), platform game levels (Compton and Mateas 2006) and game rules (Togelius and Schmidhuber 2008).

Until now, similar methodologies used for player satisfaction capture and optimization have been concentrating on the impact of NPC behavior (Yannakakis and Hallam 2007) and the adjustment of NPC internal controls for maximizing satisfaction in games (Yannakakis and Hallam 2008b). The work we describe here is concerned with the construction

---

Copyright © 2009, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.



Figure 1: Test-bed game screenshot.

of computational models of player experience derived from gameplay interaction which can be used as fitness functions for game content generation.

## Test-bed Platform game

The test-bed platform game used for our studies is a version of *Infinite Mario Bros* (see Figure 1) which is a public domain clone of Nintendo's classic platform game *Super Mario Bros*. The game is playable on the web, where Java source code is also available<sup>1</sup>. While implementing most features of the original game, its standout feature is the **automatic generation of levels**. Every time a new game is started, levels are randomly generated by traversing a fixed width and adding features (such as blocks, holes and enemies) according to certain heuristics. In our *Infinite Mario Bros* version most of the randomised placement of level features is fixed and deterministic since we concentrate on **a few selected game level parameters that affect game experience**.

## Collecting Player Data

In order to model the relation between variable features of the game and aspects of player experience, these features

---

<sup>1</sup><http://www.mojang.com/notch/mario/>

first need to be defined, and empirical data needs to be collected. We decided to focus on features which were common to most, if not all, platform games: holes and direction of movement.

### Parameterizable level generation

We modified the level generator to create levels according to four controllable parameters presented below. Three of these parameters deal with the number, width and placement of holes. The fourth parameter turns a new function, the direction switch, on or off.

- The number of holes in the level.
- The average width of holes.
- The entropy of holes with respect to which part of the level they appear in (high entropy means uniform distribution, low entropy means that they are mostly in one part of the level).
- Number of direction switches. No direction switch means that the player needs to move from left to right in order to complete the level, as in the original Super Mario Bros. If one or more direction switch is present, the level will suddenly be mirrored at random points, forcing the player to turn around and go the other way, until reaching the end of the level or the next direction switch.

Two states (low and high) for each of the four controllable parameters above are investigated. This results in  $2^4 = 16$  different variants of the game.

### Experimental methodology

We designed a game survey study to solicit pairwise emotional preferences of subjects playing different variants of the test-bed game by following the experimental protocol proposed in (Yannakakis and Hallam 2008a). Each subject plays a predefined set of four games in pairs. For each of the two pairs of games *A* and *B*, subjects report their preference for several emotional states (e.g. fun) using a 4-alternative forced choice (4-AFC) protocol.

Several statistical features are extracted from playing data which are logged during gameplay and include game completion time, time spent on various tasks (e.g. jumping, running), information on collected items (e.g. type and amount), killed enemies (e.g. type, amount, way of killing) and information on how the player died.

Data is collected over the Internet. Users are recruited via posts on blogs and mailing lists and directed to a web page containing an applet implementing the game and questionnaire<sup>2</sup>. We have so far collected enough data to have at least 2 preference instances for each pair of game variants ( $C_2^{16} = 120$  participants), but data collection is still in progress and new data will be used to improve our models.

### Modelling and optimizing player experience

We assumed there is an unknown function between individual playing characteristics (e.g. number of coins gathered),

<sup>2</sup>The game and questionnaire, which will be demonstrated at the conference, is available at [www.bluenight.dk/mario.php](http://www.bluenight.dk/mario.php)

controllable game level features (e.g. number of holes) and reported emotional preferences, and proceeded to try to predict the latter from the former.

Using neuroevolutionary preference learning of simple nonlinear perceptrons (Yannakakis and Hallam 2008a), we have been able to accurately predict certain player emotions from gameplay features. For example, whether the player is frustrated by the current game can be predicted with an accuracy of 88.66 looking at just four behavioral features. Predicting player emotions based only on controllable features is harder, but good accuracy can be achieved using nonlinear models such as multilayer perceptrons.

Work is currently in progress to use evolutionary algorithms to optimize the level design parameters (relating to holes and switches) for different objectives. We aim to be able to generate levels that tailor the playing experience according to the needs of the game design (e.g. a challenging level combined with a frustrating experience). The success of our optimization attempts will be validated with further user studies.

### Acknowledgments

The authors would like to thank Aki Järvinen and Markus Persson for insightful discussions.

### References

- Compton, K., and Mateas, M. 2006. Procedural level design for platform games. In *Proceedings of the Artificial Intelligence and Interactive Digital Entertainment International Conference (AIIDE)*.
- Csikszentmihalyi, M. 1990. *Flow: the Psychology of Optimal Experience*. Harper Collins.
- Koster, R. 2005. *A theory of fun for game design*. Paraglyph press.
- Nelson, M. J.; Ashmore, C.; and Mateas, M. 2006. Authoring an interactive narrative with declarative optimization-based drama management. In *Proceedings of the Artificial Intelligence and Interactive Digital Entertainment International Conference (AIIDE)*.
- Togelius, J., and Schmidhuber, J. 2008. An experiment in automatic game design. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games*.
- Togelius, J.; De Nardi, R.; and Lucas, S. M. 2007. Towards automatic personalised content creation in racing games. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games*.
- Yannakakis, G. N., and Hallam, J. 2007. Towards optimizing entertainment in computer games. *Applied Artificial Intelligence* 21:933–971.
- Yannakakis, G. N., and Hallam, J. 2008a. Entertainment modeling through physiology in physical play. *International Journal of Human-Computer Studies* 66:741–755.
- Yannakakis, G. N., and Hallam, J. 2008b. Real-time Adaptation of Augmented-Reality Games for Optimizing Player Satisfaction. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games*, 103–110. Perth, Australia: IEEE.

# A Temporal Data-Driven Player Model for Dynamic Difficulty Adjustment

Alexander E. Zook and Mark O. Riedl

School of Interactive Computing, College of Computing  
Georgia Institute of Technology  
[{a.zook,riedl}@gatech.edu](mailto:{a.zook,riedl}@gatech.edu)

## Abstract

Many computer games of all genres pit the player against a succession of increasingly difficult challenges such as combat with computer-controlled enemies and puzzles. Part of the fun of computer games is to master the skills necessary to complete the game. Challenge tailoring is the problem of matching the difficulty of skill-based events over the course of a game to a specific player's abilities. We present a tensor factorization approach to predicting player performance in skill-based computer games. Our tensor factorization approach is data-driven and can predict changes in players' skill mastery over time, allowing more accurate tailoring of challenges. We demonstrate the efficacy and scalability of tensor factorization models through an empirical study of human players in a simple role-playing combat game. We further find a significant correlation between these performance ratings and player subjective experiences of difficulty and discuss ways our model can be used to optimize player enjoyment.

## Introduction

Many computer games of all genres pit the player against a succession of increasingly difficult challenges: combat with NPC enemies, puzzles, strategic planning and execution, etc. The player is expected to master a set of skills pertaining to the game mechanic over the course of the game. Some believe that this mastery of the game is a fundamental aspect to having fun in computer games (Koster 2005). However, contemporary computer games are being played by increasingly diverse audiences that differ in their skills and interests in games. This increasing variability in ability, speed of mastery, and growing diversity in tastes for game aesthetic and narrative content has prompted a recent growth of interest in automated methods to fit game content to these diverse abilities and interests. These efforts require both modeling the abilities and interests of players as well as adapting existing game content to those differences.

Many games revolve around *skill-based events*, periods of game play, such as combat or puzzles, in which the player must perform a specific skill. We see two main challenges to adapting computer games to fit individual player differences: *challenge tailoring* and *challenge contextualization*.

Copyright © 2012, Association for the Advancement of Artificial Intelligence ([www.aaai.org](http://www.aaai.org)). All rights reserved.

Challenge tailoring (CT) is the problem of matching the difficulty of skill-based events over the course of a game to a specific player's abilities. For example, in an action role-playing game such as *The Legend of Zelda* challenge tailoring may manifest as configuring the number, health, or damage dealt by various enemies at various times throughout the game. CT is similar to *Dynamic Difficulty Adjustment* (DDA), which only applies to online, real-time changes to game mechanics to balance difficulty. In contrast, CT generalizes DDA to both online and offline optimization of game content and is not limited to adapting game difficulty. Challenge contextualization (CC) is the related problem of constructing game events that set up the conditions for skill events and motivate their occurrence to the player. For example, the challenge of slaying a dragon may be contextualized by the dragon kidnapping a princess. Challenge contextualization includes common AI problems of quest generation, story generation in games, and interactive storytelling.

In this paper, we focus on the challenge tailoring problem in adventure role-playing games. Realizing challenge tailoring requires both a player model and an algorithm to adapt content based on that model. Effective player modeling for the purposes of challenge tailoring requires a data-driven approach that is able to predict player behavior in situations that may have never been observed. Because players are expected to master skills over time when playing a game, the player model must also account for temporal changes in player behavior, rather than assume the player remains fixed. Modeling the temporal dynamics of a player enables an adaptive game to more effectively forecast future player behavior, accommodate those changes, and better direct players toward content they are expected to enjoy. Further, forecasting enables player models to account for interrelations among sequences of experiences—accounting for how foreshadowing may set up a better future revelation or how encountering one set of challenges builds player abilities to overcome related challenges that build off of those.

In this paper we present and evaluate a temporal player modeling approach. We employ tensor factorization techniques to create temporal models of objective player game performance over time in a turn-based role-playing game and demonstrate the efficacy of our approach over related data-driven methods through comparisons from an empirical study. We model performance instead of difficulty be-

cause performance is objectively measurable while difficulty is subjective; we show difficulty and performance are significantly correlated for this particular domain. Finally, we suggest how our tensor factorization player model may be used for challenge contextualization.

## Related Work

Smith et al. (2011) overview the landscape of player modeling in computer games. In their taxonomy, we are investigating individual, empirical, generative player models. Research in player modeling has typically addressed the challenge tailoring problem either by developing purely behavioral models or relying on predictions that ignore temporal changes in player data. Hunicke and Chapman (2004) model players by computing the average and variance of player damage and item inventory. Dynamic Difficulty Adjustment is achieved via a hand-crafted policy prescribing actions to take based on player health and inventory states. Magerko et al. (2006) interactive story players using a vector of competence levels for various skills and associated confidence values. The system selects from a pool of challenges based on a best fit between the characteristics of the challenge event and the current state of the skill vector. van Lankveld et al. (2008) role-playing game players using their progress and health, dynamically adjusting sets of presented enemies to enforce a given level of player health over progress. In contrast, our data-driven modeling approach explicitly forecasts changes in player performance, combines information across players, and proactively constructs a long-term set of challenges based on these predictions.

Subjective self-report indications of challenge have also been used to dynamically tailor game play (Yannakakis and Togelius 2011). Pedersen et al. (2009) train a neural network to predict player self-reported experiences of fun, challenge, and frustration based on measures of player behavior and in-game content. Yannakakis, Lund, and Hallam (2006) employ a neural network to predict player self-reported interest. Our approach extends these models by correlating time-varying measures of performance to self-report measures, enabling forecasts of player experience forward in time.

While we believe our work is the first application of tensor factorization to challenge tailoring problems, we note that similar techniques have been used to model student performance over time on standardized tests (Thai-Nghe, Horvath, and Schmidt-Thieme 2011).

## Player Model

We explore tensor factorization techniques for modeling player performance in action role-playing games. While we focus on action role-playing games, we believe our techniques generalize to any games that make regular use of skill-based events. Tensor factorization techniques decompose multidimensional measurements into latent components that capture underlying features of the high-dimensional data. Tensors generalize matrices, moving from the two-dimensional structure of a matrix to a three or more dimensional structure. For our player modeling approach we extend two-dimensional matrices representing player perfor-

mance against particular enemy types to add a third dimension representing the time of that performance measure.

We chose to use tensor factorization due to its favorable scaling properties, ability to cope with missing data, high accuracy, speed in generating predictions, and previous success in other applications. Tensor factorization is an extension of matrix factorization, which has been used in collaborative filtering applications—such as the Netflix Prize data mining competition—to great success. Matrix factorization offers the key advantage of leveraging information from a group of users that has experienced a set of content to make predictions for what a new group of individuals that has only been partially exposed to that content will do. Specifically, user data is represented in a  $M = U \times I$  matrix indicating user preference ratings on items and decomposition extracts latent factors relating to users and items. Tensor factorization adds more dimensions, such as time, and extracts latent factors related to these other dimensions as well. Matrix and tensor factorization scale effectively to large numbers of users and items, handle missing information from users and achieve high accuracy (Koren and Bell 2011; Su and Khoshgoftaar 2009).

Formally, we represent player data in a tensor  $Z = U \times I \times T$ . In our simple spell-casting action role-playing game (described in the next section),  $U$  is the player (“user”),  $I$  is the spell type (“item”) and  $T$  is the time of the performance recording. CP decomposition is a generalization of singular value decomposition from matrices to tensors. In CP decomposition the three-dimensional  $Z$  tensor is decomposed into a weighted combination of three vector latent factors,

$$Z \approx \sum_{k=1}^K \lambda_k w_k \circ h_k \circ q_k$$

where  $\circ$  is the vector outer product,  $\lambda_k$  are positive weights on the factors,  $w_k$  are player factors,  $h_k$  are spell type factors, and  $q_k$  are time factors.  $K$  is the number of components used for each factor as an approximation of the true structure, keeping the set of the  $K$  most important components found (Kolda and Bader 2009). The decomposition can be computed by minimizing the root mean squared error between the factor inner product above and true data values, iteratively fitting each of the factors while fixing values of the other factors until convergence. We employ the N-way toolbox (Andersson and Bro 2000) to perform this decomposition. Predictions in this model consist of taking the inner product of these three factors, a computationally efficient process.

## Experiment

We focus on skill-based aspects of games that require procedural knowledge: the ability to correctly act in given circumstances, typically with limited time for decision-making. To test our tensor factorization model we conducted a study of player learning in a turn-based role-playing game. Below we present the game used for the study, study methodology, and the results of comparing our modeling technique to related approaches.

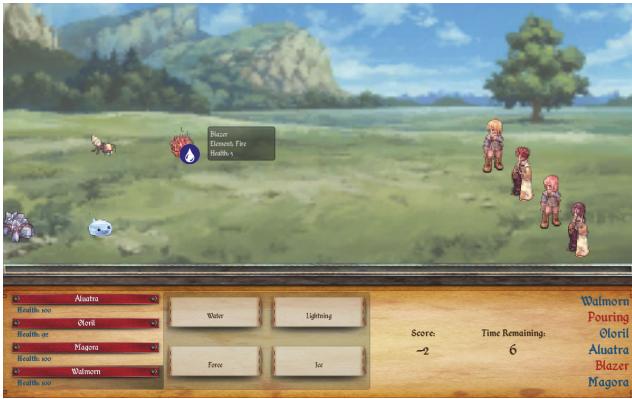


Figure 1: A battle between monsters and the player team.

Table 1: Spell Effectiveness Matrix.

| Attack ↓ Def. → | fire | water | acid | ice | light. | earth | force | undeath |
|-----------------|------|-------|------|-----|--------|-------|-------|---------|
| fire            | 1    | 0     | 1    | 2   | 1      | 2     | 1     | 0       |
| water           | 2    | 1     | 0    | 1   | 0      | 1     | 2     | 1       |
| acid            | 1    | 2     | 1    | 0   | 1      | 0     | 1     | 2       |
| ice             | 0    | 1     | 2    | 1   | 2      | 1     | 0     | 1       |
| lightning       | 1    | 2     | 1    | 0   | 1      | 0     | 1     | 2       |
| earth           | 0    | 1     | 2    | 1   | 2      | 1     | 0     | 1       |
| force           | 1    | 0     | 1    | 2   | 1      | 2     | 1     | 0       |
| undeath         | 2    | 1     | 0    | 1   | 0      | 1     | 2     | 1       |

## Game Domain

We implemented a turn-based role-playing game in which the player leads a group of four characters through a sequence of spell-casting battles against groups of enemy monsters. For the purpose of this study, we ignore the quest-like contextualization of the battles. See Figure 1 for the game’s battle interface. Turns are limited to 10 seconds to require mastery of a complex spell system.

Each enemy is associated with one of eight spell types and player-controlled characters can attack with four of the eight possible spells. Casting a particular spell against an enemy of a particular type results in an attack being effective, ineffective, or super-effective, resulting in normal damage, no damage, or double damage against an enemy (see Table 1).

We intentionally created a spell system that was difficult to completely memorize, but contained intuitive combinations—water spells are super-effective against fire enemies—and unintuitive combinations—undead spells are super-effective against force enemies—ensuring that skill mastery could only be achieved by playing the game. Note that pairs of spells—e.g., fire and force—are repeated in Table 1. This ensures a simpler underlying skill structure for players to learn; there are effectively only four spells. A scoring system based on spell effectiveness motivates players to learn; effective spells earn two points, ineffective spells earn zero points, and super-effective spells earn five points. Enemy attacks decrease player score by one. Player characters were assigned different spell sets, forcing players to learn the spell system.

## Study Methodology

We recruited 32 participants to play our role-playing game, which was simplified to present a series of battles one after another, as in Figure 1. In our study we first gave players five minutes to review a text document explaining the spell system and the game interface. Once familiar with the game, players completed the sequence of eleven battles while we recorded the performance of the player in terms of effectiveness of spells chosen against enemies. After each battle we additionally asked players to report how difficult and enjoyable the battle was on a 5-point Likert scale.

We recorded each spell players cast on each enemy on each turn and battle in the game along with the associated performance value: 0 for ineffective, 1 for effective, and 2 for super-effective. Because spell effectiveness determines damage to enemies, player behavior traces varied in the number of turns taken, but had the same number of battles. We average performance for each spell type across all turns within a given battle, leaving the performance value missing for any spells not used in the battle.

We hypothesize that a tensor factorization approach will outperform non-tensor approaches. The tensor factorization model organizes player data in a three-dimensional tensor (player, enemy spell type, battle number), where each point in the tensor is the average performance of the player in attacking foes of a given spell type during a given battle. Spell types not used in a given battle were recorded as missing values. This produces a  $32 \times 8 \times 11$  tensor for our 32 players, 8 spell types, and 11 battles. We compared the tensor model to two other models: matrix factorization using an unfolded tensor, and matrix factorization averaging over time. The matrix unfolding of this tensor simply concatenates battles column-wise, producing a  $32 \times 88$  matrix recording player performance on each spell type and battle number combination. The final matrix factorization approach averaged player performance against spell types across all battles, removing any temporal information. Data points represent average player performance over their entire battle history against an enemy type. If our hypothesis is confirmed, then the tensor model captures additional structure lost in the unfolded matrix model when all time points are concatenated together.

We also hypothesize that there is an inverse relationship between objective, measurable player performance and subjective, self-reported difficulty. Should this hypothesis hold it will verify that in the context of action role-playing games we can use skill performance as a proxy for difficulty.

## Results

We first compared players according to a variety of collected demographic information (age, gender, race, ethnicity, prior video game and role-playing game experience) and found no significant differences among these groups in our data set, validating the use of a single model across all players. We measured the 10-fold cross-validated *percent variance explained* of the tensor, matrix unfolded, and time-averaged models. Percent variance explained is an error measure that compares the total variation in the data across cases with the variation remaining after a model has been applied to the

data. It describes how well a model captures variations in the data, with higher percentages indicating more powerful explanations.

The tensor model outperforms the other techniques for three or more components (see Table 2). While the tensor model does not achieve substantially better performance than the unfolded or time-averaged models on few components it shows much greater performance on larger numbers of components, reflecting its greater capacity to model complex underlying structure in the data. Notably, the tensor model shows comparable high performance for three, four, or five factors, indicating the spell matrix reflects an underlying structure testing the corresponding number of skills. As noted earlier our spell matrix actually only contains four spells (each spell has two names). This intuitively explains a result suggesting four underlying factors—the four unique spells—with similar numbers of factors reflecting a moderate amount of noise around this underlying information. The 100% score for the time-averaged model with 8 factors reflects the fact that this model is modeling 8 spell types with 8 factors and thus can trivially recover the same model.

| components | tensor | unfolded | timeavg |
|------------|--------|----------|---------|
| 2          | 92.59  | 90.82    | 95.86   |
| 3          | 92.18  | 89.44    | 72.41   |
| 4          | 88.72  | 86.30    | 39.99   |
| 5          | 90.11  | 61.77    | 45.02   |
| 6          | 89.11  | 63.66    | 24.28   |
| 7          | 87.11  | 24.29    | 36.30   |
| 8          | 80.05  | -10.81   | 100.00  |

Table 2: Comparison of percent variance explained by different models using varying numbers of factors.

We evaluated the scaling of the tensor model in three ways: (1) how well the model scales with additional data; (2) how well the model forecasts into the future for players; and (3) how well the model scales in the number of players used by the system. In all three cases we use 10-fold cross-validated percent variance explained averaged over 10 repetitions of the experiment. Our first assessment hid a random subset of all our data and generated predictions for those missing values as shown in Figure 2. Model performance increased with the amount of total data used, with the simple two-component model showing high performance across all amounts of data while the more complex six- and eight-component models required approximately 60% of the data to achieve high performance. Thus, tensor factorization techniques perform well with this kind and amount of data, with more complex models requiring additional data but achieving high performance with more information.

Our second assessment evaluated the accuracy of the tensor model predictions on future events by hiding all future battles for a group of target players (Figure 3). The simple two-component tensor model achieved high performance regardless of the number of battles used, while the more complex six- and eight-component models required approximately seven battles to achieve comparable performance.

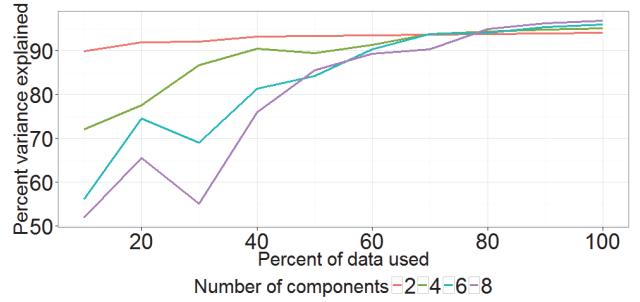


Figure 2: Percent variance explained of the tensor model as a function of the amount of data randomly subsampled from the data set, averaged over 10 repetitions.

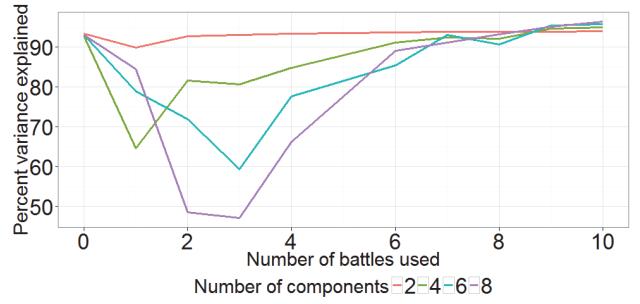


Figure 3: Percent variance explained of the tensor model when using first subset of battle data from randomly selected players, averaged over 10 repetitions.

Thus, tensor factorization models can generate useful forecasts after only a small number of battles observed for any given player. From a game design perspective, this suggests a relatively short training level in which the player can demonstrate skills can yield reasonable predictive power.

Our third assessment examined the tensor model accuracy when training and testing on varying numbers of players. We randomly subsampled from our full data set to use 6, 11, 16, 21, or 27 players and performed the same cross-validation analysis as above. Accuracy improved on the 2 component model from 81.17% variance explained with 6 players to 86.52% with 27 players, while the 4 component model improved from 52.55% to 86.29%, respectively. Other numbers of components showed similar trends, converging to approximately 86% accuracy with 27 players. The results of these three assessments demonstrate the power of the tensor factorization technique to scale with additional players and additional data for those players.

Finally, we found a significant correlation between objectively measure performance and subjectively experienced difficulty. Performance levels significantly differed (Kruskal-Wallis test for analysis of variance,  $p < 0.001$ ) between subjective difficulty ratings. A Kruskal-Wallis test assesses whether responses (here performance) in different groups (difficulty ratings) shows significant differences—it is a non-parametric alternative to the standard analysis of variance tests that assume the data has a Gaussian distri-

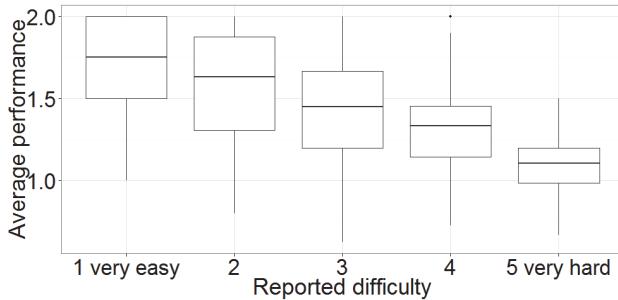


Figure 4: Comparison between objective performance values for subjective difficulty ratings, showing median values and the range from 25<sup>th</sup> and 75<sup>th</sup> percentiles.

bution, as our data showed skewing in performance measures for different response groups. A Dunnett’s test found all adjacent pairs of difficulty ratings significantly differed ( $p < 0.05$ ) in performance value, ranging from  $-0.11$  to  $-0.22$  (Figure 4). Dunnett’s test applies a more powerful version of the Student’s t-test to performance multiple comparisons among groups. Thus our second hypothesis is confirmed; objective measures of skill performance are inversely related to perceived difficulty.

## Content Adaptation

Our temporal player model can predict player performance on skill-based events in the future. To perform challenge tailoring of a game, the player model must be combined with information about how to use the model. In the next sections we describe (1) a target expected performance to compare predictions of an individual user’s performance against, and (2) an algorithm to select and/or parameterize skill-based events to bring predicted player performance in line with target performance.

## Target Performance Curve

We expand upon the concept of a *performance curve* (Zook et al. 2012), as an indication of the desired player performance over a sequence of skill-based challenges. In our game this indicates desired player performance across a sequence of battles, provided by the human designer. Figure 5 shows an example of a performance curve. This particular curve seeks a decrease in performance over time until the very end of the game. This game should appear to a player to increase in difficulty from challenge to challenge, even as the player continues to master the skills involved. Other curves are possible as well. A curve expressed by  $p = c$  (a horizontal line at a fixed constant,  $c$ ) indicates a game in which the difficulty appears to remain the same, even as the player’s skills improve. That is, the challenges may be parameterized to be more difficult, but because the challenge level is increasing at the same rate as player skill mastery, there should be little or no perceived change in difficulty. More complicated patterns, such as a series of rises and falls, can express complex designer intentions.

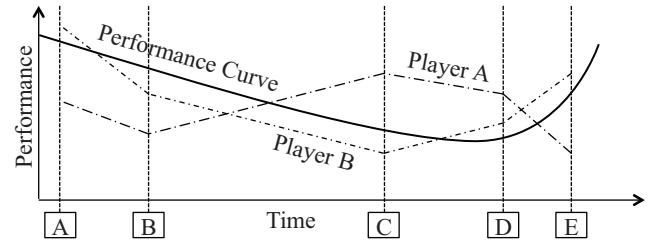


Figure 5: An example performance curve indicating a game that gets progressively more difficult until the very end, even as the player’s skills improve. Dotted lines indicated predicted performance for two players.

Note that performance curves are specifications of desired player performance, rather than difficulty. Although our studies show that performance and difficulty are inversely correlated, there may be other aspects of the game—such as ambiguity in game information—that are not necessarily captured in the performance metric. A performance curve bypasses the ambiguity of subjective difficulty while enabling designers to specify how they desire a particular performance metric to vary over the course of an experience.

## Challenge Tailoring and Contextualization

Given a performance curve and a temporal player model, challenge tailoring is the problem of selecting and spacing a sequence of skill-based events—in our case, battles—that match the predicted performance over time to the desired performance. This is a discrete optimization problem—battles are discrete units and the goal is to minimize the difference between predicted and desired performance values for these battles. A variety of techniques may be applied to solve these problems including constraint satisfaction, dynamic programming, and heuristic search techniques such as genetic algorithms (Smith and Mateas 2011; Togelius et al. 2011; Sorenson, Pasquier, and DiPaola 2011; Zook et al. 2012). In contrast to the reactive, near-term changes typically employed in DDA (Magerko, Stensrud, and Holt 2006; Hunnicke and Chapman 2004), temporal player models are able to also proactively restructure long-term content to optimize a global player experience.

Challenge contextualization is the problem of selecting non-skill-based events that explain, motivate, or justify the skill-based events. Quest generation and narrative generation techniques may be used to provide this contextualization of skill-based events, although other, non-narrative contextualization may exist as well. Challenge tailoring and challenge contextualization may be performed in a pipelined fashion or in parallel. Pipeline techniques afford modular and compositional approaches to tailoring game content. In particular, if tailoring of skill-based events takes precedence over contextualization such that contextualization (i.e., the narrative, quest, or mission) can be sub-optimal then one might choose to solve the discrete optimization problem of selecting and spacing all skill-based events before “filling the gaps” with non-skill-based, contextualizing events (cf., (Magerko, Stensrud, and Holt 2006)). Fixing the skill-based

events prior to contextualization makes challenge tailoring easier, but constrains the searchable context space.

Parallel techniques, conversely, may explore the full space of joint skill- and non-skill combinations, but trade this flexibility for greater complexity in the tailoring task. For example, we describe a technique that searches for a complete sequence of skill- and non-skill-based events that simultaneously optimizes for player performance and context (Zook et al. 2012). The question of whether to use a pipeline versus parallel approach depends on the importance of contextualization relative to skill tailoring and the extent to which skill- and non-skill-based events appear seamless.

## Conclusions

As computer games grow in popularity personalization of game content—which we cast as the paired problems of challenge tailoring and challenge contextualization—will also grow in importance. Unlike many other player modeling approaches to challenge tailoring and dynamic difficulty adjustment, we use a data driven approach that explicitly incorporates a temporal component. This temporal component allows us to more accurately forecast *future* player performance by modeling changes in a player’s skills.

A performance curve provides authorial control over the game in the form of a target level of performance. Since objective performance is inversely correlated with subjective difficulty, a performance curve can guide an algorithm in the selection and parameterization of skill-based events over time. The tensor factorization model gives a system insight into how the player will perform on various sequences of challenges. With these tools, there are many discrete-optimization algorithms that can solve the challenge tailoring problem. Challenge contextualization, though outside the scope of this paper, can further ensure challenges make sense to the player while promoting player skill mastery.

Mastery of skills is correlated with fun (Koster 2005). A game that is able to tailor its difficulty to meet the abilities of the player may be perceived as more enjoyable to a wider range of players because it is never unintentionally boringly easy or frustratingly hard. This in turn has the potential to increase game replayability and make certain games accessible to a wider diversity of players.

## Acknowledgments

The project or effort described here has been sponsored by the U.S. Army Research, Development, and Engineering Command (RDECOM). Statements and opinions expressed do not necessarily reflect the position or the policy of the United States Government, and no official endorsement should be inferred.

## References

- Andersson, C., and Bro, R. 2000. The N-way toolbox for MATLAB. *Chemometrics and Intelligent Laboratory Systems* 52(1):1–4.
- Hunicke, R., and Chapman, V. 2004. AI for dynamic difficulty adjustment in games. In *Proceedings of the AAAI Workshop on Challenges in Game Artificial Intelligence*.
- Kolda, T., and Bader, B. 2009. Tensor decompositions and applications. *SIAM review* 51(3):455–500.
- Koren, Y., and Bell, R. 2011. Advances in Collaborative Filtering. In Ricci, F.; Rokach, L.; Shapira, B.; and Kantor, P. B., eds., *Recommender Systems Handbook*. Boston, MA: Springer. 145–186.
- Koster, R. 2005. *A Theory of Fun in Game Design*. Paraglyph press.
- Magerko, B.; Stensrud, B.; and Holt, L. 2006. Bringing the schoolhouse inside the box - a tool for engaging, individualized training. In *Proceedings of the 25th Army Science Conference*.
- Pedersen, C.; Togelius, J.; and Yannakakis, G. N. 2009. Modeling Player Experience in Super Mario Bros. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games*.
- Smith, A., and Mateas, M. 2011. Answer set programming for procedural content generation: A design space approach. *IEEE Transactions on Computational Intelligence and AI in Games* 3(3):187–200.
- Smith, A.; Lewis, C.; Hullett, K.; Smith, G.; and Sullivan, A. 2011. An inclusive view of player modeling. In *Proceedings of the 6th International Conference on Foundations of Digital Games*.
- Sorenson, N.; Pasquier, P.; and DiPaola, S. 2011. A Generic Approach to Challenge Modeling for the Procedural Creation of Video Game Levels. *IEEE Transactions on Computational Intelligence and AI in Games* 3(3):229–244.
- Su, X., and Khoshgoftaar, T. M. 2009. A Survey of Collaborative Filtering Techniques. *Advances in Artificial Intelligence* 2009:1–19.
- Thai-Nghe, N.; Horvath, T.; and Schmidt-Thieme, L. 2011. Factorization Models for Forecasting Student Performance. In *Proceedings of the 4th International Conference on Educational Data Mining*.
- Togelius, J.; Yannakakis, G.; Stanley, K.; and Browne, C. 2011. Search-Based Procedural Content Generation: A Taxonomy and Survey. *IEEE Transactions on Computational Intelligence and AI in Games* 3(3):172–186.
- van Lankveld, G.; Spronck, P.; and Rauterberg, M. 2008. Difficulty scaling through incongruity. In *Proceedings of the 4th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*.
- Yannakakis, G. N., and Togelius, J. 2011. Experience-Driven Procedural Content Generation. *IEEE Transactions on Affective Computing* 2:147–161.
- Yannakakis, G.; Lund, H.; and Hallam, J. 2006. Modeling children’s entertainment in the playware playground. In *Proceedings of the 2nd IEEE Symposium on Computational Intelligence and Games*.
- Zook, A.; Riedl, M. O.; Holden, H. K.; Sotilare, R. A.; and Brawner, K. W. 2012. Automated scenario generation: Toward tailored and optimized military training in virtual environments. In *Proceedings of the 7th International Conference on the Foundations of Digital Games*.

# Polymorph: Dynamic Difficulty Adjustment Through Level Generation

Martin Jennings-Teats, Gillian Smith, Noah Wardrip-Fruin

Expressive Intelligence Studio  
University of California, Santa Cruz  
Santa Cruz, CA, USA

{mjennin1, gsmith, nwf}@soe.ucsc.edu

## ABSTRACT

Players begin games at different skill levels and develop their skill at different rates so that even the best-designed games are uninterestingly easy for some players and frustratingly difficult for others. A proposed answer to this challenge is Dynamic Difficulty Adjustment (DDA), a general category of approaches that alter games during play, in response to player performance. However, nearly all these techniques are focused on basic parameter tweaking, while the difficulty of many games is connected to aspects that are more challenging to adjust dynamically, such as level design. Further, most DDA techniques are based on designer intuition, which may not reflect actual play patterns. Responding to these challenges, we present Polymorph, which employs techniques from level generation and machine learning to understand game component difficulty and player skill, dynamically constructing a 2D platformer game with continually-appropriate challenge. We believe this will create a play experience that is unique because the changes are both personalized and structural, while also providing an example of a promising new research and development approach.

## Categories and Subject Descriptors

K.8.0 [Personal Computing]: General – Games. I.2.6 [Artificial Intelligence]: Learning.

## General Terms

Design, Human Factors.

## Keywords

Games, level design, dynamic difficulty adjustment, procedural content generation.

## 1. INTRODUCTION

The classic 2D side-scrolling platformer is a genre of games that focuses on jumping dexterity and precise timing to get past obstacles in fairly linear levels; for example, Super Mario Bros [8]. The game levels are designed to be difficult and unforgiving, so the player is only able to complete a level after playing it partway through multiple times to learn the exact necessary pattern of actions. This genre of game has been very popular, but

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

PCGames 2010, June 18, Monterey, CA, USA

Copyright 2010 ACM 978-1-4503-0023-0/10/06... \$10.00

it cannot be said to cater to every player's experience and abilities. This is one example of the types of problems that can be addressed with Dynamic Difficulty Adjustment (DDA).

This paper describes the vision and implementation of Polymorph. The goal of Polymorph is to automatically generate 2D platformer levels during play as a means of dynamic difficulty adjustment. Specifically, rather than being authored by hand, game levels will be procedurally generated as the player moves through the level, one chunk at a time as needed. The generation of these chunks will be customized to match the player's performance, so that each player will be presented with a level that provides a challenge appropriate to their skill. This is not to say that the player will never die in a tough section or breeze through an easy section, but the game will correct for this in the next section, hopefully avoiding difficulty-related player frustration and boredom and providing an example of a promising new approach to DDA.

We tackle the DDA problem by creating a statistical model of difficulty in 2D platformer levels along with a model of the player's current skill level. These models are gleaned with machine learning techniques from play traces collected with a game-like tool. The models are used to select the appropriate level segment for a player's current performance. The level segments are generated automatically using a variation on the work of [14], which is described in more detail in section 2.2.

This paper shows how a game can be designed to accommodate the skill and experience of every individual player by incorporating machine learning techniques and dynamic level generation. This is an advance on prior work in dynamic difficulty adjustment, which has for the most part avoided adaptive level design, and in procedural level generation, which has mainly focused on creating full levels for replayability. Polymorph is a work in progress: a data collection tool, the level generator, the game engine, and a pilot study have been completed.

## 2. RELATED WORK

### 2.1 Dynamic Difficulty Adjustment

Game designers nearly always strive to create games in which the difficulty of the obstacles presented to the player is appropriate for the player's skill level. As a player's skill improves through practice, a well designed game will present more formidable difficulties so that the player is never bored by overly easy gameplay or frustrated by overly difficult gameplay [3] [6]. This in itself is a very challenging design task however, and game designers spend great effort making sure that their game is well balanced so that challenges will be appropriate for players' abilities. Even so, designers are usually not able to accommodate

every player's skill level, and frustrating mismatches between a player's skill and a game's difficulty are common [5] [12].

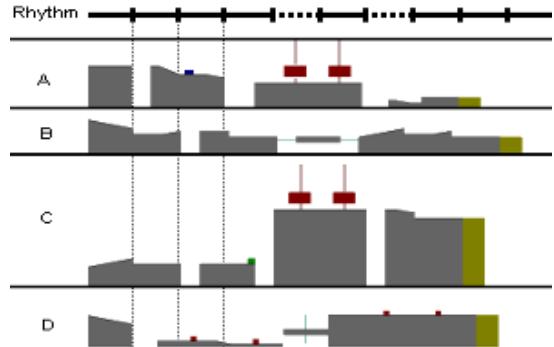
As described briefly above, **Dynamic Difficulty Adjustment (DDA)** is a term for techniques in which games automatically alter themselves in some way to better fit the skill levels of the players. This is common in the genre of racing games with the practice of “rubber banding”, wherein players in last place are granted an increased maximum speed [5]. DDA is rarely used in other game genres, but there are some notable exceptions. One of the most complex examples of DDA in a commercial game is the first-person shooter SiN Episodes. It uses a statistical model of player performance with “advisor” sub-systems that adjust attributes such as the number of concurrent enemies, the damage and accuracy of the enemies' weapons, and the enemies' tendency toward throwing grenades [7] [13]. Hunicke created the Hamlet system, which uses sets of probabilities to determine the appropriate time to intervene in a first-person shooter by giving the player more ammo or a health boost [5]. What these DDA strategies all have in common is that the intervention into the game is primarily through a numeric attribute adjustment. In contrast, the dynamic changes made by Polymorph are structural rather than numeric in nature.

An alternative method of intervention, on which this paper is focused, is the modification of the level design. For example, Left 4 Dead changes the location and frequency of spawn points for enemies and items based on player performance, which can have a significant impact on player experience but is not a substantial change in the structural design of the levels [2] [18]. Pedersen et al. created a version of Infinite Super Mario Bros from which they derived a statistical model of player challenge and frustration, among other emotional states [11]. Using their evolutionary algorithms, this model could be used to generate levels for a particular level of player challenge, which is the closest work (of which we are aware) to the approach of Polymorph [10]. However, it is not designed to be dynamic during play, unlike Polymorph, which generates sections of a level ahead of the player's movement, allowing a level to change in difficulty from start to finish in response to changes in the player's performance.

## 2.2 Procedural Level Generation

Procedural level generation has been used in games for decades, with popular RPGs such as Rogue and Diablo, as well as some 2D platformers such as Spelunky [1] [17] [19]. These games typically work by fitting together hand-authored level chunks into random combinations. Pedersen et al.'s variation on Infinite Super Mario Bros also works on this principle, combining level chunks to create a level that is measured according to a statistical model of the emotions it would evoke in a player [10]. This work, along with Togelius et al.'s 2007 work on generating tracks for racing games, uses an evolutionary algorithm to iteratively create similar levels with slight modifications [15]. These approaches differ from Polymorph by generating geometry in larger granularity, as well as in Polymorph's unique learning features (see section 4).

Smith et al. created a generator for 2D platformer levels based on a model of player action-rhythm, which is the basis for the level generation done in this project. This approach starts with a rhythm



**Figure 1. Several possible mappings of geometry onto rhythm.**

of desired player actions, such as run, jump or wait. The generator then chooses from sets of geometry that can fulfill each of these actions—the same starting rhythm can produce many distinct level designs depending on the geometry selected for each action, as shown in figure 1. Because the generation is based on player actions and their associated level geometry, it has a finer granularity of control over the level design than the previously mentioned techniques which use hand-authored level chunks [14]. All of these strategies for level generation have been offline full-level generation techniques, meaning that they create an entire playable level as a whole—usually ahead of time, rather than generating parts of the level during play [16]. This is because a primary motivation for procedural level generation has been to create improved replayability for a game, which can be accomplished by giving the player a new level each time through. This is an effective strategy for creating engaging game experiences, but online level generation, in which the player's behavior alters the level as they play, is a much more dynamic approach to the core challenge to which Polymorph responds: difficulty adjustment.

The game Charbitat is an example of online, real-time level generation, where the player's preference for interacting with certain elements will alter the game world to increase the prevalence of that element [9]. This focus on the world's elements differs substantially from Polymorph's focus on difficulty.

## 3. DATA COLLECTION MECHANISM

### 3.1 Tool

In order to generate parts of a level to match a player's skill level, we need both a model of difficulty in our domain of 2D platformer levels and a dynamic model of the player's current performance. To answer these two questions—what makes a 2D platformer level difficult or easy, and how do we determine if a player is struggling or needs more of a challenge—we turn to a strategy of **mass data collection** and statistical machine learning. We created a data collection tool that asks a human player to play a short (approximately 10 seconds) level segment, collecting data on the level and the player's behavior along the way. The collected data and its use as machine learning features are discussed in more depth in section 4. After the player completes the level or their character dies, they are asked to label the level segment by answering the multiple choice question: **how difficult was this level segment?** The label choices presented to the player are *1-Easy* through *6-Hard*. Only data from players completing multiple levels is considered in order to avoid subjective difficulty ratings.

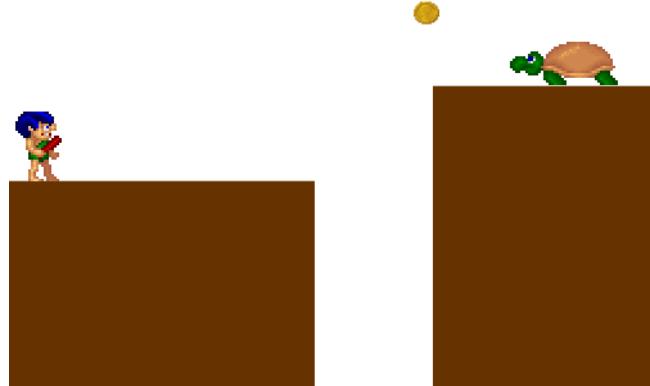
The level segments are generated by an adaptation of the action rhythm-based generator from [14], described briefly in section 2.2. We do not put any restrictions on the rhythms used to generate level segments, since we want to consider all playable segments. We believe that difficulty in interesting 2D platformer levels comes largely from the combination of adjacent components and not just from the presence or absence of a particular component (see section 4). This belief is the reason we limit the level segments to such a short length. With short level segments, which don't contain too many level components, we attempt to control which independent variables, in this case level component interactions, might be resulting in the difficulty label the player assigned to the segment. We recognize that not all aspects of level challenge are captured by these short segments, but we are focusing on the micro level of component combinations rather than level-wide patterns or the introduction of new mechanics.

One potential drawback to this method of data collection is that a player might not have a good understanding of the game mechanics their first time through a short segment and might therefore rate it as more difficult than they would after they gained more experience. However, we believe that this is representative of playing a full game, where the player learns the game and increases in skill as they progress, so that a player participating in our data collection over many level segments will help us to model difficulty for an average player. Also, this short level segment seems ideal as the amount of granularity for custom, player performance-based, generated level chunks as the player progresses through a level of the final game. Each time the player successfully passes through a segment of this length (or dies), another segment of the same length will be generated and placed in front of them.

The tool is Flash-based so that it can be easily distributed and used through most web-browsers by many simultaneous players. We have completed a pilot study with more than one hundred undergraduate game design students from UC Santa Cruz, allowing us to refine our instruments and the machine learning features discussed in section 4. The preliminary data and resulting refinements have been encouraging for the goal of the data collection: to model player performance and difficulty in 2D platformer levels. We are currently preparing to distribute the tool much more widely to collect data from thousands of participants over multiple level segment playthroughs.

### 3.2 Generate and Test

One of the early challenges in the development of Polymorph was the tendency of the level generation algorithm to create many more easy level segments than difficult level segments, though the generator is capable of expressing levels across a wide range of difficulty. This was due to the many possible rhythm-geometry mappings that do not present a challenge for the player—a flat surface with several short gaps, for instance. This was problematic for the data collection mechanism, since the players assigned labels for low difficulty levels far more often than for high difficulty levels. The distribution of the data was therefore skewed toward the low-difficulty end of the spectrum.



**Figure 2.** Part of a level segment in Polymorph's data collection tool with the player character on the left. The segment includes a jump up over a gap, a coin and a moving enemy.

The distribution of level segments has been corrected by creating several heuristic critic modules. When a level segment is first generated, each of these critics will estimate its difficulty on a particular metric. For example, one critic examines the level segment's action-density, while another simply counts the number of potentially deadly level components present, since some components do not create the possibility of player death on their own. Once all of the critics have examined the level segment, it is classified into several estimated difficulty categories, which the data collection tool samples for segments presented to the player rather than choosing a random unplayed level segment, thus modifying the distribution of segments so that it will be more spread across the range of level difficulty. The critics only decide which segments to show to players and are not considered for the final ratings of challenge.

## 4. LEARNING FEATURES

The first statistical model that we want Polymorph to learn from the collected data is a ranking of level segments according to their difficulty. As mentioned previously, we believe that difficulty in 2D platformer levels is related to the combinations of adjacent level components more than to the presence of a particular level component. Using the example shown in figure 2, a gap by itself is easy to overcome and a slow plodding enemy is not much of a difficulty, but by placing the enemy on the landing platform of the gap the level designer has created a much larger challenge for the player, requiring more exact timing and prediction of the enemy's movements. Therefore we have included as learning features not only the number of occurrences of a particular level component, but also the occurrences of any two-component adjacency in the level segment. Using the example depicted in figure 2 once again, the feature regarding the number of upward-rising gaps in the segment is incremented, as is the feature regarding the number of gap-enemy adjacencies. Other level segment-related features of interest include the average gap width, the total change in altitude of the platforms in the level and the width of the largest and smallest platforms.

Polymorph also needs a statistical model of the player's current skill level. The data collection tool keeps track of features representing the player's behavior while playing. Some of the more interesting features are the amount of time the player spends standing still or moving backwards, the total completion time of

the level segment, the number of coins collected, and whether the player died or completed the segment. The data collection tool does not ask the player how well they think they were performing, but we assume that this is implicit in their answer to how difficult they think the level segment was.

Given the level-descriptive features we will apply a machine-learned ranking algorithm such as Ranking SVM to rank all of the level segments generated during play of the final game [4]. Meanwhile, Polymorph will be collecting the player behavior features, which will be evaluated on a model trained with the data from the collection tool. Then, before the player progresses into the next segment of the level a new segment from the ranked list will be chosen according to the model of the player's current skill level. This way, as the player learns to play the game better and improves their personal skill, the level will increase in difficulty to compensate and maintain an appropriate challenge. Alternatively, if it becomes clear that the player is struggling, the next segment of the level will be chosen to reduce challenge.

## 5. CONCLUSION & FUTURE WORK

We have described the vision, completed work, and further plans for dynamic difficulty adjustment in the game Polymorph. The player-specific adjustment is achieved by procedurally generating the level during play. At this time we have created the game engine, the level generator, and the data collection tool, as well as run a pilot study. To complete the game we will collect data on a much larger scale with the online tool, and we will process the data as discussed in section 4 to create models of level difficulty and player performance. Final game polish will be applied, with commissioned artwork and tweaking of the mechanics based on evaluative playtests.

The largest challenge for the development of Polymorph has been the task of designing features to collect from the generation of level segments. The features need to be broad enough to represent the difficulty of a level for an average player while remaining specific enough to be generalizable to other, very different level designs. We are confident in the features we have chosen, but we do not claim to have created a perfect model of difficulty in 2D platformer levels. Considering the pair-wise adjacencies of level components will help to address the problem of difficulty arising from the interaction of level components rather than from the presence of individual components. However, in the analysis of challenging hand-authored platformer levels it seems that the interaction of more than two components is common. Creating features to represent these more complicated interaction settings would be an improvement and is a direction we would like to pursue with future iterations of Polymorph.

Procedurally generating level segments online, in real-time as a method of dynamic difficulty adjustment allows for intervention that is both a structural change and personalized to the player's skill and experience. We believe this will give Polymorph a unique play experience and demonstrate the strength of combining techniques from level generation and machine learning for dynamic difficulty adjustment.

## 6. REFERENCES

[1] Blizzard Entertainment 1997. Diablo.

- [2] Booth, M. 2009. The AI Systems of Left 4 Dead. Keynote, Fifth Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE '09). Stanford, CA. October 14 – 16, 2009.
- [3] Fullerton, T., Swain, C., and Hoffman, S. 2004. Improving player choices. Gamasutra (March 2004). [http://www.gamasutra.com/features/20040310/fullerton\\_01.shtml](http://www.gamasutra.com/features/20040310/fullerton_01.shtml). Online Feb. 1, 2005.
- [4] Herbrich, R., Graepel, T., and Obermayer, K. 2000. Large Margin Rank Boundaries for Ordinal Regression. Advances in Large Margin Classifiers, 115-132, Liu Press.
- [5] Hunicke, R. 2005. The case for dynamic difficulty adjustment in games. In Proceedings of the 2005 ACM SIGCHI international Conference on Advances in Computer Entertainment Technology (Valencia, Spain, June 15 - 17, 2005). ACE '05, vol. 265. ACM, New York, NY, 429-433.
- [6] Juul, J. 2009. Fear of Failing? The Many Meanings of Difficulty in Video Games. The Video Game Theory Reader 2, B. Perron and M. Wolf, Ed. Routledge, London.
- [7] Kazemi, D. 2008. Metrics and Dynamic Difficulty in Ritual's SiN Episodes. OrbusGameWorks.com. <http://orbusgameworks.com/blog/article/70/metrics-and-dynamic-difficulty-in-rituals-sin-episodes-part-1>
- [8] Nintendo 1985. Super Mario Bros.
- [9] Nitsche, M., Ashmore, C., Hankinson, W., Fitzpatrick, R., Kelly, J., and Margenau, K. 2006. Designing Procedural Game Spaces: A Case Study. In Proceedings of FuturePlay 2006. London, Ontario. October 10 – 12, 2006.
- [10] Pedersen, C., Togelius, J., and Yannakakis, G. 2009. Modeling Player Experience in Super Mario Bros. Proceedings of the 2009 IEEE Symposium on Computational Intelligence and Games (Politecnico di Milano, Milano, Italy, September 07-10, 2009).
- [11] Persson, M. Infinite Mario Bros.
- [12] Phillips, B. 2009. Staying Power: Rethinking Feedback to Keep Players in the Game. Gamasutra.com. [http://www.gamasutra.com/view/feature/4171/staying\\_power\\_rethinking\\_feedback\\_.php](http://www.gamasutra.com/view/feature/4171/staying_power_rethinking_feedback_.php)
- [13] Ritual Entertainment 1998. SiN Episodes.
- [14] Smith, G., Treanor, M., Whitehead, J., Mateas, M. 2009. Rhythm-Based Level Generation for 2D Platformers. Proceedings of the 2009 Int'l Conference on the Foundations of Digital Games (Orlando, FL, USA, April 26-30, 2009).
- [15] Togelius, J., De Nardi, R., and Lucas, S. 2007. Towards automatic personalised content creation for racing games. Proceedings of the 2007 IEEE Symposium on Computational Intelligence and Games (2007).
- [16] Togelius, J., Yannakakis, G., Stanley, K., and Browne, C. 2010. Search-based Procedural Content Generation. To be presented at Evostar (Istanbul Technical University, Istanbul, Turkey, April 07-09, 2010).
- [17] Toy, M. and Wichman, G. 1980. Rogue.
- [18] Valve Software. 2008. Left 4 Dead.
- [19] Yu, D. 2009. Spelunky.

# Game AI Revisited

Georgios N. Yannakakis  
Center for Computer Games Research  
IT University of Copenhagen  
Rued Langgaards Vej 7  
Copenhagen, Denmark  
yannakakis@itu.dk

## ABSTRACT

More than a decade after the early research efforts on the use of artificial intelligence (AI) in computer games and the establishment of a new AI domain the term “game AI” needs to be redefined. Traditionally, the tasks associated with game AI revolved around non player character (NPC) behavior at different levels of control, varying from navigation and pathfinding to decision making. Commercial-standard games developed over the last 15 years and current game productions, however, suggest that the traditional challenges of game AI have been well addressed via the use of sophisticated AI approaches, not necessarily following or inspired by advances in academic practices. The **marginal penetration** of traditional academic game AI methods in industrial productions has been mainly due to the lack of constructive communication between academia and industry in the early days of academic game AI, and the inability of academic game AI to propose methods that would significantly advance existing development processes or provide scalable solutions to real world problems. Recently, however, there has been a shift of research focus as the current plethora of AI uses in games is breaking the non-player character AI tradition. A number of those alternative AI uses have already shown a significant potential for the design of better games.

This paper presents four key game AI research areas that are currently reshaping the research roadmap in the game AI field and evidently put the game AI term under a new perspective. These game AI flagship research areas include the computational modeling of player experience, the procedural generation of content, the mining of player data on massive-scale and the alternative AI research foci for enhancing NPC capabilities.

## Categories and Subject Descriptors

I.2.1 [Artificial Intelligence]: Applications and Expert Systems—*Games*; H.1.2 [Models and Principles]: User—Machine Systems—*Human factors*

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CF'12, May 15–17, 2012, Cagliari, Italy.

Copyright 2012 ACM 978-1-4503-1215-8/12/05 ...\$10.00.

## Keywords

Game artificial intelligence, player experience modeling, procedural content generation, game data mining, game AI flagships

## 1. INTRODUCTION

Almost 30 years after the first reported video game conference at Harvard [33] and 12 years after Laird's and van Lent's seminal article [26] that, in part, established the foundations of game artificial intelligence (AI) and inspired early work in the field [34, 22, 25, 14, 3, 30, 58] the game AI term needs to be revisited and restructured.

Since those first days of academic game AI the term was mainly linked to non player character (NPC) behavior (i.e. NPC AI) and pathfinding [8] as most of the early work in that field was conducted by researchers with AI, optimization and control background and research experience in adaptive behavior, robotics and multi-agent systems<sup>1</sup>. AI academics used the best of their computational intelligence and AI tools to enhance NPC behavior in generally simple, research-focused, non-scalable projects of low commercial value and perspective. In almost every occasion the two (academic and industrial game AI), rather immature, communities would meet they would conclude about the gap existent between them and the need of bridging it for their mutual benefit [8]. The key message of academic AI has been that industry does not attempt to use sophisticated AI techniques with high potential (e.g. neural networks) in their games. On the other end, the central complaint of industrial game AI has been the lack of domain-knowledge and practical wisdom when it comes to realistic problems and challenges faced during game production.

While the vast majority of AI academics (including the author) would claim that games are fully scripted and still use 30-year old AI technology — such as A\* and finite state machines — the game industry had been making small, yet important, steps towards integrating nouvelle (or modern) AI [8] in their games [55] during the early days of game AI. A non-inclusive list of games that advanced the game AI state-of-practice in industry [42] includes the advanced sensory system of guards in *Thief* (EIDOS, 1989); the advanced opponent tactics in *Half-Life* (Valve, 1998); the fusion of ma-

<sup>1</sup>Note that this paper deliberately excludes research in board game AI as — in contrast to the breadth and multifaceted nature of AI research challenges met in game development — advances in that field can only be algorithmic with respect to a particular aim (i.e. learn to play a board game) in constrained board game spaces.

chine learning techniques such as perceptrons, decision trees and reinforcement learning coupled with the belief-desire-intention cognitive model in *Black and White* (EA, 2000); the dynamic difficulty adjustment (DDA) features in the *Halo* series (MS Game Studios); the imitation learning *Dri-vatar* system of *Forza Motorsport* (MS Game Studios, 2005); the AI director of *Left 4 Dead* (Valve, 2008)<sup>2</sup> and the neuroevolutionary training of platoons in *Supreme Commander 2* (Square Enix, 2010).

The key criterion that distinguishes a successful AI in commercial-standard games had always been the level of integration and interweaving of AI in the design of the game [42]. While an unsuccessful coupling of game design and AI may lead to unjustifiable NPC behaviors, break the suspension of disbelief and immediately reduce player incorporation [6], the successful integration of AI in the design process in games such as *Façade* [31] or *Kinectimals* (MS Game Studios, 2010) may absorb potential “catastrophic” failures or limitations of the AI.

The level of AI sophistication in recent games such as *Left 4 Dead* (Valve, 2008) and *The Elder Scrolls V: Skyrim* (Bethesda Softworks, 2011) suggests that advances in NPC AI have converged to highly satisfactory solutions for most NPC control challenges faced during game production. Moreover, a number of game developers (and some game AI academics) have already taken sides arguing that NPC AI is almost solved [7, 35] for most production tasks while some claim that game AI research and development should focus on non-traditional uses of AI [35, 45]. Such indications suggest that further marginal enhancements of NPC AI may require significant effort and cost.

Due to the rise of robust and effective industrial game AI solutions, more frequent and constructive communication with the industry, the convergence to satisfying NPC performances, the support of the multidisciplinary nature of game AI and a more pragmatic and holistic view of the game AI problem, recent years have seen a shift of academic interests with respect to game AI. We have reached an era where the catholic focus of the application of AI in the domain of games is not on agents and NPC behaviors. The focus has, instead, started to shift towards interweaving game design and game technology by viewing the role of AI holistically: AI can help us to make better games but that does not necessarily imply better, more human-like or believable NPCs.

There are a number of key research areas, which I name *game AI flagships*, that have recently provided innovative, yet commercially-plausible solutions for a number of game development challenges. Those areas of common (academic and industrial) interest appear to both synthesize the framework of current and future academic research and already influence high-end commercial game technology. It is expected that a focus on these game AI areas (beyond NPC control) will most likely yield a larger impact on the making of better games via the use of AI. Player Experience Modeling (PEM), Procedural Content Generation (PCG), Large-Scale Game Data Mining and new perspectives in NPC AI are the four main game AI flagships considered in this paper. The list provided in this paper is, by no means, inclusive of all high-end potential game AI areas but it is representa-

<sup>2</sup>The success of the AI director and its positive impact to player experience has influenced game AI architectures in a number of other game productions including *Resistance 3* (Insomniac Games, 2011).

tive of spotlight current research efforts and development advances.

## 2. THE FLAGSHIPS OF GAME AI

In this section the emerging, non-traditional, flagship research areas of game AI are presented, corresponding successful examples are provided for each flagship, and arguments are listed for their inclusion as key game AI research and development areas.

### 2.1 Player Experience Modeling

Recent years have seen both a boost in the size of the gaming population and a demographic diversification of computer game players [23]. This, in turn, means that skills, preferences and experience differ widely among players of the same game. Therefore, the need for tailoring games to individual playing experiences is growing and the tasks of user modeling and experience-based adaptation within games become increasingly important and challenging. Game engines that are able to recognize and model the playing style and detect the current emotional and cognitive state of the user will be necessary milestones towards the personalization of the playing experience.

Player experience modeling (PEM) is the study and use of AI techniques for the construction of computational models of experience of players. PEM places an AI umbrella to the multidisciplinary intersection of the fields of user (player) modeling, affective computing, experimental psychology and human-computer interaction. Player experience, player satisfaction and their modeling have recently seen a growing number of dedicated workshops, special sessions and invited talks in top academic venues including the IEEE Conference on Computational Intelligence and Games (IEEE-CIG)<sup>3</sup>, the Foundations of Digital Games (FDG)<sup>4</sup> and the Artificial Intelligence and Interactive Digital Entertainment (AIIDE) conference<sup>5</sup> and special issues to journals such as the IEEE Transactions of Computational Intelligence and AI in Games and IEEE Transactions on Affective Computing. In addition, top game developers (such as Valve) have recently started to experiment with multiple modalities of user input (e.g. physiology) for the personalization of experience in popular games such as *Left 4 Dead* (Valve, 2008) [1].

#### 2.1.1 General PEM Principles

A model of player experience predicts some aspect of the experience of a player in general, a type of player, or a particular player would have in some game situation. There are many ways this can be achieved, with approaches to PEM varying both regarding the inputs (from what the experience is predicted, e.g. physiology, level design parameters, playing style or game speed), outputs (what sort of experience is predicted, e.g. fun, frustration, attention or immersion) and the modeling methodology.

Computational models of player experience can be built on different types of data collected from the players which in turn define different approaches to player experience modeling (PEM). We can identify three main classes of approaches for modeling player experience in games which rely on (1) data expressed by players (*subjective PEM*); (2) player data

<sup>3</sup>[www.ieee-cig.org/](http://www.ieee-cig.org/)

<sup>4</sup>[www.foundationsofdigitalgames.org/](http://www.foundationsofdigitalgames.org/)

<sup>5</sup><http://www.aiide.org/>

obtained from alternative modalities of player response (*objective* PEM); and (3) contextual and behavioral data obtained through the interaction between the player and the game (*gameplay-based* PEM). Data from multiple modalities and types can be fused to better predict annotated player experience states.

If data recorded includes a scalar representation of experience, or classes and annotated labels of user (cognitive and affective) states any of a large number of machine learning (regression and classification) algorithms can be used to build models of experience. Available methods include neural networks, Bayesian networks, decision trees, support vector machines and standard linear regression. On the other hand, if experience is annotated in a ranking format (e.g. game version X is more frustrating than game version Y) standard supervised learning techniques are inapplicable, as the problem becomes one of *preference learning* [15, 57]. In particular, neuro-evolutionary preference learning has proven suitable for this task; in this method, the weights of neural networks are evolved to minimize the error between reported and predicted preferences [63, 57].

The following subsections provide further details about each of the three PEM approaches and corresponding successful examples of each approach. The section ends with a discussion on the potential of personalization of both the experience and the player experience model.

### 2.1.2 Subjective PEM

The most direct way to develop a model of experience is to ask the players themselves about their playing experience and build a model based on such data. Subjective PEM considers first person reports (*self-reports*). Reports expressed indirectly by experts or external observers can potentially provide reliable player experience annotations; however, third-person assessment is not covered in this paper. Subjective player experience modeling can be based on either players' *free-response* during play or on *forced* data retrieved through questionnaires. Forced self-reports can be further classified as *ratings*, in which the players are asked to answer questionnaire items given in a *Likert scale* or *rankings*, in which players are asked to compare their player experience in two or more sessions of the game [60, 57, 51]. A recent study has exposed the limitations of rating approaches over ranking questionnaire schemes (e.g. pairwise preference) including increased order of play and inconsistency effects [56].

While self-reports have inherent limitations including user self-deception, memory-dependencies and ordering effects numerous studies have shown that ranked self-reporting can successfully guide machine learning algorithms to capture aspects of player experience in prey/predator [59], physical interactive [61], platform [41, 40] and racing [51] games.

### 2.1.3 Objective PEM

Player experience can be linked to a stream of emotions, which may be active simultaneously, usually triggered by events occurring during gameplay. Games can elicit player emotional responses which in turn may affect changes in the player's physiology [64, 51], reflect on the player's facial expression [39, 24], posture and speech, and alter the player's attention and focus level [2]. Monitoring such bodily alterations may assist in recognizing and synthesizing the emotional responses of the player. The *objective* approach to

player experience modeling incorporates access to multiple modalities of player input for the purpose of modeling the affective state of the player during play.

Models built via the objective PEM approach may be very accurate representations of player experience since player experience is approached in a holistic manner via the use of multiple input modalities. The key limitations of the objective PEM approach include its high intrusiveness and questionable feasibility. Most modalities are still nowadays not technically plausible within commercial computer games. For instance, existing hardware for physiology requires the placement of body parts (e.g. head, chest or fingertips) to the sensors making physiological signals such as EEG, respiration, blood volume pulse and skin conductance rather impractical and highly intrusive for most games. However, recent advances on biofeedback sensor technology have resulted in low-cost, unobtrusive biofeedback devices (bracelet sensors) appropriate for gaming applications<sup>6</sup>.

Pupilometry and gaze tracking are very sensitive to distance from screen and variations in light and screen luminescence, which makes them rather impractical for use in a game application. Modalities such as facial expression and speech could be technically plausible in games even though the majority of the vision-based affect-detection systems currently available cannot operate in real-time [67]. At the positive end of the spectrum, Microsoft's XBox 360 Kinect<sup>7</sup> sensor device is pointing towards more natural game interaction and showcases a promising future of objective PEM.

### 2.1.4 Gameplay-based PEM

The main assumption that drives *gameplay-based* PEM is that player actions and real-time preferences are linked to player experience since games may affect the player's cognitive processing patterns and cognitive focus. On the same basis, cognitive processes may influence emotions as one may infer the player's emotional state by analyzing patterns of the interaction and associating user emotions with context variables. Any element derived from the interaction between the player and the game forms the basis for gameplay-based PEM. This includes parameters from the player's behavior derived from responses to system elements.

The inputs to a gameplay-based player experience model are statistical spatio-temporal features of game interaction. Those features are usually mapped to levels of cognitive states such as attention, challenge and engagement [11]. General measures such as performance and time spent on a task have been used in the literature, but also game-specific measures such as the weapons selected in a shooter game [18]. Moreover, several dissimilar difficulty and challenge measures (see [21, 37, 52] among many) have been proposed for different game genres. In all of these studies, difficulty adjustment is performed, based on a player experience model that implies a direct link between challenge and player satisfaction. Sometimes a player model [62, 20, 10] is embedded in the process of PEM. Data mining attempts to predict player actions and intentions as well as to identify different playing patterns within a game [12, 53] can also be viewed as gameplay-based PEM. Game data mining is covered in Section 2.3 in further detail as it is considered a game AI flagship on its own.

<sup>6</sup><http://www.emoticalab.com/>

<sup>7</sup><http://www.xbox.com/kinect/>

Gameplay-based PEM is arguably the most computationally efficient and least intrusive PEM approach but it usually results in a low-resolution model of playing experience.

### 2.1.5 Personalizing PEM

AI methodology can be used not only to construct a computational model of player experience but to also tailor the player experience model itself to the player’s individual preferences during the interaction. An example of this promising direction within PEM research is the work of Liapis et al. [28] where computational models of player aesthetics are tailored to the player’s selections and are further used for the design of personalized spaceships with respect to player aesthetics (see Fig. 2).

## 2.2 Procedural Content Generation

Procedural content generation (PCG) can be viewed as the study and development of algorithms that generate content automatically. *Game content* refers to all adjustable game elements that may affect player experience (excluding NPC behavior) which may include elements such as terrains, maps, levels, stories, quests, rulesets, camera profiles and music. There are several benefits obtained from the automatic creation of content in games [50]: first, PCG can alleviate the enormous effort and cost of content creation and make it easier to tailor content to the player; second, content can automatically adapt the game to the needs and preferences of individual players and yield maximal game replayability; third, PCG can challenge human creativity and generate solutions beyond the designer’s imagination in a stand-alone or mixed-initiative design [44, 4] fashion.

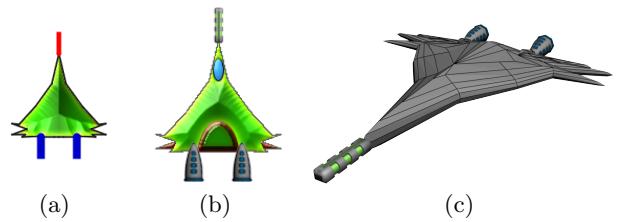
Even though PCG techniques have been incorporated in games since *Rogue* (1980) it is only very recently that an academic community is devoted to the study of PCG signaling the shift of interest towards this use of AI in games. That trend is reflected by an IEEE CIS Task Force<sup>8</sup> and a wiki<sup>9</sup> on the topic, a series of dedicated workshops at the FDG conference, an international PCG competition<sup>10</sup> and a special issue on PCG at the IEEE Transactions of Computational Intelligence and AI in Games. The use of PCG for the design of better games has reached a peak of interest in commercial game development which is showcased by successful (almost entirely procedurally generated) games such as *Minecraft* (Mojang, 2011) and *Love* (Eskil Steenberg, 2010) and the broad coverage of PCG topics in relevant conferences (such as the Paris Game AI conference series).

Research efforts that couple the PEM and the PCG flagships has resulted to research projects of high commercial potential under the *experience-driven procedural content generation* (EDPCG) framework [65]. According to the EDPCG framework, content is viewed as a building block of player experience which can be adjusted to optimize the experience of the player (predicted via player experience models). Examples of EDPCG work include the adaptive content creation framework of Shaker et al. [43] where personalized Super Mario Levels are generated for maximizing models of player experience states, such as *fun*, which are built via crowd-sourced *fun* reports about mini Super Mario Bros levels (see Fig. 1 for two example levels).

<sup>8</sup><http://game.itu.dk/pcg/>

<sup>9</sup><http://pcg.wikidot.com>

<sup>10</sup><http://www.marioai.org>



**Figure 2:** Example spaceship (rendered with three different methods) generated via an EDPCG algorithm. The algorithm both tailors computational user aesthetics models and generates personalized spaceships based on those tailored models.

In addition to Super Mario Bros levels, racing tracks [47], strategy maps [48], game rule sets [5], buildings [29] and weapons [17] (among other types of content) have been generated based on models of player experience. The work of Liapis et al. [27, 28] is indicative of the power of EDPCG for game design as personalized spaceships can rapidly be generated based on player aesthetics models via interactive evolution. Both the models of user aesthetics and the aesthetic attributes of the spaceships are adapted to the preferences of the user/designer yielding personalized spaceship designs such as those presented in Fig. 2.

## 2.3 Massive-Scale Game Data Mining

Game data mining may be loosely defined as the use of AI (data mining algorithms) for addressing questions such as: *how do people play a game?*; *is the game played as intended?*; *why do people stop playing a game?*; *why do we play a game this way?*; *can we predict what a player will do?*; *does the game offer the right experience?*; *what is the personality of a player?*. All these are critical questions that are tied to user-oriented testing procedures used in the game industry. In iterative-phased game development, representative samples of the target audience as well as internal professional testers spend time and put effort on testing the games and evaluating the quality of the gaming experience.

During the last five years — as an alternative to traditional testing — key game developers (including Zynga, Blizzard, Bioware, Square Enix Europe and EA Games) have been collecting and analyzing detailed and massive-scale player behavioral and contextual data (i.e. game metrics) via specialized monitoring software. As argued by big data analysts we have now reached a point where existing data mining algorithms cannot follow the growth of data availability and the massive size of datasets available and, thereby, cannot fully support the analysis of such data. This poses new exciting challenges and avenues of research for AI in games since the use of AI for inferring playing patterns from data can provide a quantitative approach to and supplement of traditional qualitative approaches of user and playability testing [13].

Even though directly linked to context-based PEM (see Section 2.1), the mining of gameplay data deserves its own game AI flagship as game metrics and game metric analysis is currently a spotlight research and development area within the games industry supported by a growing number of game data analytics companies. Game data mining has seen extensive coverage in game developer meetings such as

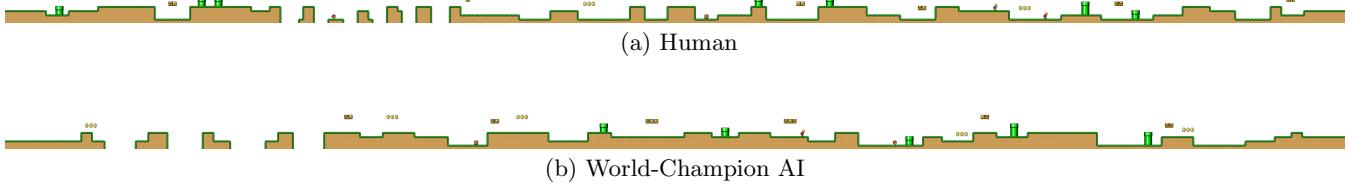


Figure 1: Example levels generated for two different Super Mario players. The levels generated maximize the modeled *fun* value for each player. The level on top depicts the level generated for a human player while the level below is the level generated for the world champion agent of the Mario AI competition.

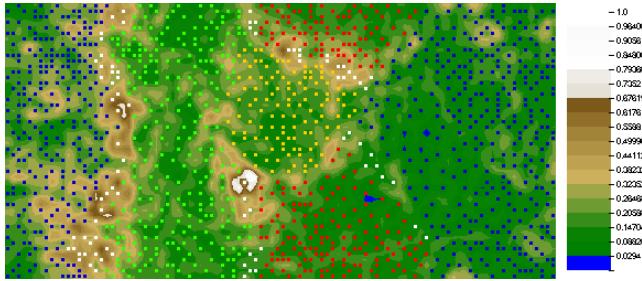


Figure 3: U-matrix visualization of a self-organizing map depicting the 4 player clusters identified in a population of 1365 Tomb Raider: Underworld players (shown as small colored squares). Different square colors depict different player clusters. Valleys represent clusters whereas mountains represent cluster borders.

the game AI summit at GDC<sup>11</sup> and the Paris Game AI Conference<sup>12</sup> as well as dedicated panels, tutorials and special sessions in top game AI academic conferences such as IEEE-CIG and AIIDE.

Among the relatively few studies in the young field of game data mining [13], Yee has analyzed the relationship between player motivations, demographic variables and in-game behaviors of 3000 MMORPG players [66]. Drachen et al. [12] have identified four potential player types in *Tomb Raider: Underworld* using self-organization (see Fig. 3) in direct collaboration with the developer of the game (i.e. Crystal Dynamics). Thurau et al. [46] have applied non-negative-matrix factorization to mine 1.6 million images on World of Warcraft guilds while Mateas and Weber [54] have mined game metrical data for the prediction of player strategies in StarCraft. In addition to empirical player data, alternative analytical approaches have been proposed for evaluating games and their playability [36].

## 2.4 NPC AI: Different Perspectives

As AI has already provided satisfactory solutions to most NPC tasks (including navigation and lower levels of NPC control) the focus of research on NPC AI may shift towards under-researched, yet very promising, directions that will enhance NPC capabilities. A different perspective to NPC AI is to view NPC control as a mapping of the NPC’s context (environment) and attempt to alter the latter to observe

changes in the perception of the first. So far, the question of whether empirical research efforts should be put more on the agent or its environment (or both) in order for the agent to appear more believable, human-like, or intelligent remains largely unanswered. The ability of the environment — instead of, or in addition to NPC attributes — to absorb non-believable agent behaviors can define new variables for optimization. This raises new research questions such as *how can the design of a game be altered to allow for maximal absorption of AI weaknesses with minimal effort and how can constructive or search-based [50] content creation processes be coupled with NPC AI control for achieving such a goal*. The issue of assessing NPC believability through contextual content creation and adaptation has already been addressed by a recent study on the believability of Super Mario Bros players [49]. In addition, game Turing test competitions such as those in Super Mario Bros<sup>13</sup> and in *Unreal Tournament* (Epic Games, 1999) [19] define attempts on further exploring the unknown mapping between NPC agent behavior, game context and NPC believability.

Beyond standard single NPC control, a promising trend on NPC AI research — which already has an impact on recent game productions — appears to be the generation and detection of patterns of complex social behavior and interaction among NPCs and humans [68, 38] with a focus on cognitive/affective agent architectures for social games such as the *Prom Week* game [32]. In addition, data-driven modeling of groups of NPCs and players via group structure identification [16] can offer a complementary perspective towards well-grounded human behavior models [9] that can guide personalization in social games.

## 3. CONCLUSIONS

More than ten years after the establishment of the game AI field the term needs to be revisited and enhanced with non-traditional research and development areas beyond NPC control. The plethora of ways AI is currently used in games, beyond traditional areas such as NPC AI, showcases the potential and impact of a broader conception of the research field, and can enlarge the boundaries of design within these creative industries.

This paper listed a number of flagship areas that are currently at the spotlight of game AI state-of-the-art research and commercial-standard development. Methods for **modelling player experience**, algorithms and processes for generating content of high value automatically, approaches for mining massive-scale data of players and alternate perspec-

<sup>11</sup><http://www.gdconf.com/>

<sup>12</sup><http://gameaiconf.com/>

<sup>13</sup><http://www.marioai.org/turing-test-track>

tives on NPC AI research define the framework of the four key game AI areas presented.

The list of flagships is not inclusive of all potential core uses of AI in the years to follow. In addition to the game AI flagships discussed in this paper the current trends of pervasiveness, embedded systems and natural interaction in design have already seen their integration in gaming contexts (e.g. the Primesense camera-based sensor). Thus, natural and multimodal interaction for player behavioral and movement pattern analysis arguably define core AI domains in the near future at the crossroad of the game data mining and the player experience modeling flagships. Finally, at the crossroads of procedural content generation and player experience modeling, substantial effort is expected on the development of sophisticated AI techniques for meaningful story generation and the design of personalized authoring tools.

## 4. ACKNOWLEDGMENTS

The research was supported, in part, by the FP7 ICT project SIREN (project no: 258453) and by the Danish Research Agency, Ministry of Science, Technology and Innovation project AGameComIn; project number: 274-09-0083. Thanks to Mark J. Nelson and Julian Togelius for comments and pointers.

## 5. REFERENCES

- [1] M. Ambinder. Biofeedback in Gameplay: How Valve Measures Physiology to Enhance Gaming Experience. In *Game Developers Conference*, 2011.
- [2] S. Asteriadis, K. Karpouzis, and S. D. Kollias. A neuro-fuzzy approach to user attention recognition. In *Proceedings of ICANN*, pages 927–936. Springer, 2008.
- [3] C. Bauckhage, C. Thurau, and G. Sagerer. Learning human-like opponent behavior for interactive computer games. *Pattern Recognition, Lecture Notes in Computer Science 2781*, pages 148–155, 2003.
- [4] R. Bidarra, K. de Kraker, R. Smelik, and T. Tutenel. Integrating semantics and procedural generation: key enabling factors for declarative modeling of virtual worlds. In *Proceedings of the FOCUS K3D Conference on Semantic 3D Media and Content, France (February 2010)*, 2010.
- [5] C. Browne. *Automatic generation and evaluation of recombination games*. PhD thesis, Queensland University of Technology, 2008.
- [6] G. Calleja. *In-Game: From Immersion to Incorporation*. The MIT Press, 2011.
- [7] A. Champandard. Tutorial presentation. In *IEEE Conference on Computational Intelligence and Games*, 2012.
- [8] A. J. Champandard. *AI Game Development*. New Riders Publishing, 2004.
- [9] Y. Chang, T. Levinboim, V. Rajan, and R. Maheswaran. Learning and Evaluating Human-Like NPC Behaviors in Dynamic Games. In *Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, 2011.
- [10] D. Charles and M. Black. Dynamic player modelling: A framework for player-centric digital games. In *Proceedings of the International Conference on Computer Games: Artificial Intelligence, Design and Education*, pages 29–35, 2004.
- [11] C. Conati. Probabilistic Assessment of User’s Emotions in Educational Games. *Journal of Applied Artificial Intelligence, special issue on “Merging Cognition and Affect in HCI”*, 16:555–575, 2002.
- [12] A. Drachen, A. Canossa, and G. N. Yannakakis. Player Modeling using Self-Organization in Tomb Raider: Underworld. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games*, pages 1–8, Milan, Italy, September 2009. IEEE.
- [13] A. Drachen, C. Thurau, J. Togelius, and G. N. Yannakakis. *Game Telemetry and Metrics*, chapter Large-scale Data Mining in Games. Springer-Verlag, 2012.
- [14] M. Freed, T. Bear, H. Goldman, G. Hyatt, P. Reber, A. Sylvan, and J. Tauber. Towards more human-like computer opponents. In *Working Notes of the AAAI Spring Symposium on Artificial Intelligence and Interactive Entertainment*, pages 22–26, 2000.
- [15] J. Fürnkranz and E. Hüllermeier. Preference learning. *Künstliche Intelligenz*, 19(1):60–61, 2005.
- [16] C. Grappiolo, Y. G. Cheong, R. Khaled, and G. N. Yannakakis. Modelling Global Pattern Formations for Collaborative Learning Environments. In *Proceedings of the 12th IEEE International Conference on Advanced Learning Technologies*.
- [17] E. Hastings, R. Guha, and K. Stanley. Automatic content generation in the galactic arms race video game. *Computational Intelligence and AI in Games, IEEE Transactions on*, 1(4):245–263, 2009.
- [18] E. Hastings, R. Guha, and K. O. Stanley. Evolving content in the galactic arms race video game. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games*, pages 241–248, 2009.
- [19] P. Hingston. A Turing Test for Computer Game Bots. *IEEE Transactions on Computational Intelligence and AI in Games*, 1, 2009.
- [20] R. Houlette. *Player Modeling for Adaptive Games. AI Game Programming Wisdom II*, pages 557–566. Charles River Media, Inc, 2004.
- [21] H. Iida, N. Takeshita, and J. Yoshimura. A metric for entertainment of boardgames: its implication for evolution of chess variants. In R. Nakatsu and J. Hoshino, editors, *IWEC2002 Proceedings*, pages 65–72. Kluwer, 2003.
- [22] D. Isla and B. Blumberg. New challenges for character-based AI for games. In *Proceedings of the AAAI Spring Symposium on AI and Interactive Entertainment*, pages 41–45. AAAI Press, 2002.
- [23] J. Juul. *A Casual Revolution: Reinventing Video Games and Their Players*. MIT Press, 2009.
- [24] L. Kessous, G. Castellano, and G. Caridakis. Multimodal emotion recognition in speech-based interaction using facial expression, body gesture and acoustic analysis. *Journal on Multimodal User Interfaces*, 3:33–48, 2010.
- [25] J. E. Laird. Research in human-level AI using computer games. *Communications of the ACM*, 3(8):32–35, 2002.
- [26] J. E. Laird and M. van Lent. Human-level AI’s killer

- application: Interactive computer games. In *Proceedings of the Seventh National Conference on Artificial Intelligence (AAAI)*, pages 1171–1178, 2000.
- [27] A. Liapis, G. N. Yannakakis, and J. Togelius. Optimizing Visual Properties of Game Content Through Neuroevolution. In *Artificial Intelligence for Interactive Digital Entertainment Conference*, 2011.
- [28] A. Liapis, G. N. Yannakakis, and J. Togelius. Adapting Models of Visual Aesthetics for Personalized Content Creation. *IEEE Transactions on Computational Intelligence and AI in Games, Special Issue on Computational Aesthetics in Games*, 2012. (to appear).
- [29] A. Martin, A. Lim, S. Colton, and C. Browne. Evolving 3D buildings for the prototype video game subversion. *Applications of Evolutionary Computation*, pages 111–120, 2010.
- [30] M. Mateas. Expressive ai: Games and artificial intelligence. In *Level Up: Digital Games Research Conference*.
- [31] M. Mateas and A. Stern. Façade: An experiment in building a fully-realized interactive drama. In *Game Developers Conference Game Design track*, volume 2, 2003.
- [32] J. McCoy, M. Treanor, B. Samuel, A. Reed, M. Mateas, and N. Wardrip-Fruin. Prom Week. In *Foundations of Digital Games*, 2012.
- [33] E. Mitchell. Video games visit harvard yard. *Antic*, 2:24, 1983.
- [34] A. Nareyek. Intelligent agents for computer games. In T. Marsland and I. Frank, editors, *Computers and Games, Second International Conference, CG 2002*, pages 414–422, 2002.
- [35] A. Nareyek. Game AI is dead. Long live game AI! *IEEE Intelligent Systems*, 22(1):9–11, 2007.
- [36] M. J. Nelson. Game metrics without players: Strategies for understanding game artifacts. In *Proceedings of the 2011 AIIDE Workshop on Artificial Intelligence in the Game Design Process*, pages 14–18, 2011.
- [37] J. K. Olesen, G. N. Yannakakis, and J. Hallam. Real-time challenge balance in an RTS game using rtNEAT. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games*, pages 87–94, Perth, Australia, December 2008. IEEE.
- [38] J. Orkin, T. Smith, and D. Roy. Behavior compilation for ai in games. In *Proceedings of the Sixth Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE)*, pages 162–167, 2010.
- [39] M. Pantic and G. Caridakis. *Emotion-Oriented Systems: The Humaine Handbook*, chapter Image and Video Processing for Affective Applications, pages 101–117. Springer-Verlag Berlin Heidelberg, 2011.
- [40] C. Pedersen, J. Togelius, and G. N. Yannakakis. Modeling Player Experience in Super Mario Bros. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games*, pages 132–139, Milan, Italy, September 2009. IEEE.
- [41] C. Pedersen, J. Togelius, and G. N. Yannakakis. Modeling Player Experience for Content Creation. *IEEE Transactions on Computational Intelligence and AI in Games*, 2(1):54–67, 2010.
- [42] S. Rabin. *AI Game Programming Wisdom*. Charles River Media, Inc, 2002.
- [43] N. Shaker, G. Yannakakis, and J. Togelius. Towards automatic personalized content generation for platform games. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*. AAAI Press, 2010.
- [44] G. Smith, J. Whitehead, and M. Mateas. Tanagra: Reactive Planning and Constraint Solving for Mixed-Initiative Level Design. *Computational Intelligence and AI in Games, IEEE Transactions on*, (99):1–1, 2011.
- [45] B. Sunshine-Hill, M. Robbins, and C. Journey. Off the Beaten Path: Non-Traditional Uses of AI. In *Game Developers Conference, AI Summit*, 2012.
- [46] C. Thurau, K. Kersting, and C. Bauckhage. Convex non-negative matrix factorization in the wild. In *Proceedings of the 9th IEEE International Conference on Data Mining (ICDM'09)*, Miami, FL, USA, December 2009.
- [47] J. Togelius, R. De Nardi, and S. Lucas. Towards automatic personalised content creation for racing games. In *Computational Intelligence and Games, 2007. CIG 2007. IEEE Symposium on*, pages 252–259. IEEE, 2007.
- [48] J. Togelius, M. Preuss, N. Beume, S. Wessing, J. Hagelback, and G. Yannakakis. Multiobjective exploration of the starcraft map space. In *Computational Intelligence and Games (CIG), 2010 IEEE Symposium on*, pages 265–272. IEEE, 2010.
- [49] J. Togelius, G. Yannakakis, S. Karakovskiy, and N. Shaker. *Believability in Computer Games*, chapter Assessing Believability. Springer-Verlag, 2012.
- [50] J. Togelius, G. Yannakakis, K. Stanley, and C. Browne. Search-based Procedural Content Generation: A Taxonomy and Survey. *Computational Intelligence and AI in Games, IEEE Transactions on*, (99):1–1, 2011.
- [51] S. Tognetti, M. Garbarino, A. Bonarini, and M. Matteucci. Modeling enjoyment preference from physiological responses in a car racing game. In *Proceedings of the IEEE Conference on Computational Intelligence and Games*, pages 321–328, Copenhagen, Denmark, 18–21 August 2010.
- [52] G. van Lankveld, P. Spronck, and M. Rauterberg. Difficulty Scaling through Incongruity. In *Proceedings of the 4th International Artificial Intelligence and Interactive Digital Entertainment Conference*, pages 228–229. AAAI Press, 2008.
- [53] B. Weber and M. Mateas. A Data Mining Approach to Strategy Prediction. In *IEEE Symposium on Computational Intelligence in Games (CIG 2009)*, pages 140–147, Milan, Italy, September 2009.
- [54] B. Weber and M. Mateas. A Data Mining Approach to Strategy Prediction. In *Proceedings of the IEEE Symposium on Computational Intelligence in Games*, pages 140–147, Milan, Italy, September 2009. IEEE.
- [55] S. Woodcock. Game AI: The State of the Industry 2000-2001: It's not Just Art, It's Engineering. August 2001.
- [56] G. Yannakakis and J. Hallam. Rating vs. Preference: a comparative study of self-reporting. *Affective*

- Computing and Intelligent Interaction*, pages 437–446, 2011.
- [57] G. N. Yannakakis. Preference Learning for Affective Modeling. In *Proceedings of the Int. Conf. on Affective Computing and Intelligent Interaction*, pages 126–131, Amsterdam, The Netherlands, September 2009. IEEE.
- [58] G. N. Yannakakis and J. Hallam. Evolving Opponents for Interesting Interactive Computer Games. In S. Schaal, A. Ijspeert, A. Billard, S. Vijayakumar, J. Hallam, and J.-A. Meyer, editors, *From Animals to Animats 8: Proceedings of the 8<sup>th</sup> International Conference on Simulation of Adaptive Behavior (SAB-04)*, pages 499–508, Santa Monica, LA, CA, July 2004. The MIT Press.
- [59] G. N. Yannakakis and J. Hallam. Towards Capturing and Enhancing Entertainment in Computer Games. In *Proceedings of the 4<sup>th</sup> Hellenic Conference on Artificial Intelligence, Lecture Notes in Artificial Intelligence*, volume 3955, pages 432–442, Heraklion, Greece, May 2006. Springer-Verlag.
- [60] G. N. Yannakakis and J. Hallam. Towards Optimizing Entertainment in Computer Games. *Applied Artificial Intelligence*, 21:933–971, 2007.
- [61] G. N. Yannakakis and J. Hallam. Real-time Game Adaptation for Optimizing Player Satisfaction. *IEEE Transactions on Computational Intelligence and AI in Games*, 1(2):121–133, June 2009.
- [62] G. N. Yannakakis and M. Maragoudakis. Player modeling impact on player’s entertainment in computer games. In *Proceedings of the 10<sup>th</sup> International Conference on User Modeling; Lecture Notes in Computer Science*, volume 3538, pages 74–78, Edinburgh, 24–30 July 2005. Springer-Verlag.
- [63] G. N. Yannakakis, M. Maragoudakis, and J. Hallam. Preference Learning for Cognitive Modeling: A Case Study on Entertainment Preferences. *IEEE Systems, Man and Cybernetics; Part A: Systems and Humans*, 39(6):1165–1175, November 2009.
- [64] G. N. Yannakakis, H. P. Martínez, and A. Jhala. Towards Affective Camera Control in Games. *User Modeling and User-Adapted Interaction*, 20(4):313–340, 2010.
- [65] G. N. Yannakakis and J. Togelius. Experience-Driven Procedural Content Generation. *IEEE Transactions on Affective Computing*, 2:147–161, 2011.
- [66] N. Yee. Motivations for play in online games. *CyberPsychology & Behavior*, 9(6):772–775, 2006.
- [67] Z. Zeng, M. Pantic, G. Roisman, and T. Huang. A survey of affect recognition methods: Audio, visual, and spontaneous expressions. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 31(1):39–58, 2009.
- [68] R. Zubek and M. Lewis. Managing the Masses: Crafting AI for Online Games. In *Game Developers Conference, AI Summit*, 2012.

# How to Present Game Difficulty Choices? Exploring the Impact on Player Experience

Jan D. Smeddinck<sup>1</sup>, Regan L. Mandryk<sup>2</sup>, Max V. Birk<sup>2</sup>, Kathrin M. Gerling<sup>3</sup>,  
Dietrich Barsilowski<sup>1</sup>, Rainer Malaka<sup>1</sup>

<sup>1</sup>Digital Media Lab, TZI, University of Bremen, Germany, lastname@tzi.de

<sup>2</sup>University of Saskatchewan, Saskatoon, SK, CA, firstname.lastname@usask.ca

<sup>3</sup>University of Lincoln, Lincoln, UK, kgerling@lincoln.ac.uk

## ABSTRACT

Matching game difficulty to player ability is a crucial step toward a rewarding player experience, yet making difficulty adjustments that are effective yet unobtrusive can be challenging. This paper examines the impact of automatic and player-initiated difficulty adjustment on player experience through two studies. In the first study, 40 participants played the casual game THYFTHYF either in motion-based or sedentary mode, using menu-based, embedded, or automatic difficulty adjustment. In the second study, we created an adapted version of the commercially available game *flow* to allow us to carry out a more focused study of sedentary casual play. Results from both studies demonstrate that the type of difficulty adjustment has an impact on perceived autonomy, but other player experience measures were not affected as expected. Our findings suggest that most players express a preference for manual difficulty choices, but that overall game experience was not notably impacted by automated difficulty adjustments.

## Author Keywords

Game difficulty; player experience; game-user research; flow; dynamic difficulty adjustments; feedback.

## INTRODUCTION

Research has demonstrated a breadth of benefits of games, for example, on player cognition [22], physical health [33], and general well-being [44]. Therefore, games are now targeting broad audiences with heterogeneous expectations and abilities. Particularly in the area of serious games, researchers and designers are often addressing audiences with special needs, for example, young children [21], people with disabilities [25,28], or older adults [23]. A crucial step in this process is ensuring that games meet the needs of players to provide a positive, empowering experience. To this end, it is important to provide balanced gameplay that does not overwhelm individual players by being too challenging and that enables players of different

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.

CHI'16, May 07 - 12, 2016, San Jose, CA, USA

Copyright is held by the owner/author(s). Publication rights licensed to ACM.

ACM 978-1-4503-3362-7/16/05...\$15.00

DOI: <http://dx.doi.org/10.1145/2858036.2858574>.

abilities to play together [26]. Balancing game settings to achieve captivating experiences that can harness the full motivational potential of games is challenging, and previous work has only begun to explore this area. Manual difficulty choices, typically presented as menu settings, have long been an important element of games [11]. In addition to predefined difficulty level choices that change the base level for usually ongoing difficulty increases as a game progresses, dynamic difficulty adjustments (DDA) in games can improve gameplay [30]. A growing body of work is concerned with automated difficulty adjustments, which not only promise to reduce the burden placed on players and avoid breaking the magic circle – the special place in time and space created by a game [29,43] – but also bear the potential to influence a large amount of fine-grained variables. However, selecting matching settings for individual players is a complex problem, involving the unpredictability of human agents and the variety and interplay of game settings.

Thus, recent work suggests improvements in manual difficulty choices that can also interact with dynamic difficulty adjustment [11,12]. Based on flow theory [16], the work highlights the challenge of difficulty balancing and adjustments in games with an emphasis on the importance of enabling *personal control* whilst retaining *autonomy* and *avoiding interrupting the flow* of an activity, leading to the concept of *player-oriented difficulty choices* that are embedded within the game world [11]. While related work presents a theoretical basis for the interaction with game difficulty choices, we found no reported empirical evaluations on this topic.

Stemming from our research in the area of full-body motion-based games for health, we were interested in the question: "*Do different modes of presenting difficulty choices impact the player experience?*" The concept of player-oriented difficulty choices suggests the method of embedding the choices in a way that blends with the game world, while the most common solution employed in games on the market are difficulty settings in classic menus that may adhere roughly with the visual style of the game, but do not blend with the regular game interaction or mechanics. Games with DDA frequently offer no additional difficulty choices.

We test the hypothesis that *embedded* difficulty choices lead to a better overall player experience by retaining autonomy and control, and by avoiding interruptions of the gaming that endanger immersion or might force players out of the magic circle when compared to traditional *menu* difficulty choices or *automatic* difficulty adjustments without interaction options. Our work contributes empirical insights relating to the impact of different modalities for interacting with game difficulty choices on player experience and preferences.

## BACKGROUND

Getting the level of challenge to match the capabilities and needs of a player is a core element of good player experience. This has long been discussed in game user research [14], and the most cited psychological foundation on the balance of challenges and skills is flow theory [16].

### Flow in Games

Csikszentmihalyi [17] describes flow as state of being fully “in the zone” when engaging with an activity. This can have positive effects, such as increased motivation to perform an activity again, which may be explained through a feeling of enjoyment related to evolutionary benefits of performing certain activities [17]. The most prominent precondition that is required if flow is to be achieved is an optimal balance of risk of failure (in games e.g., losing a move, a life, a level, or the entire game) and the chance to attain the goals (in games e.g., winning some points, a bonus, a level, or the entire game) when performing an activity [41]. The entire set of conditions that are prerequisites for flow experiences are in short: clear goals, immediate feedback, match of challenges and skills, action and awareness merge, concentration on the task at hand, sense of potential and control, loss of self-consciousness, sense of time altered; resulting in an experience becoming autotelic [17]. Seeing these conditions, it is not surprising that Csikszentmihalyi frequently uses games as examples of activities that can induce flow. Related work lists explicit examples of how the facilitating factors for flow are present in games [13,31,48]. Because the balance of challenge and skills is an important precondition, and skills differ between people, it becomes clear why balancing challenges is such an important aspect of game design [41]. It also becomes clear why most games still offer manual difficulty choices in setting menus: even after iterative testing and balancing, average solutions are likely suboptimal.

### Difficulty Choices in Games

Game difficulty choices that are presented in menus with typical labels such as “easy, medium, hard” can be found even in very early and simple games. The “classic way to present difficulty choices” has arguably evolved largely as a matter of technical circumstance and the prominent use of difficulty selection menus today is arguably the result of established customs. Although manual – i.e., explicit – feedback has been used for difficulty adjustments with unusual input modalities (such as biofeedback), and has shown increased immersion and affect [34], menus with

multiple levels of one monolithic and unspecific parameter remain the most common form of difficulty choice UI.

### Dynamic Difficulty Adjustments and User-guidance

Because the difficulty of a game is often the product of many different game variables, and because it can change from one moment to the next, manual difficulty choices are not always compatible with seeking the optimal game experience. *Dynamic difficulty adjustments* (DDA) that automatically adjust difficulty based on threshold heuristics [30], or on machine learning models [18], have been explored as a reasonable alternative. Such systems have even been used in regular consumer games, such as *Half-Life 2* and *Max Payne* [1,3] and in serious games [2,47].

There are great potential benefits to DDA, such as high frequency and detailed adjustments [3,30], and in theory they merely require an adequate *performance evaluation* and an *adjustment mechanism* [1]. However, DDA adoption faces challenges, because decisions that contradict the will of the players are potentially harmful to player experience, and in the details of the implementation, assessing affective states is hard [32], especially when the tools should be unintrusive. Balancing adjustment mechanisms is extremely challenging due to the very personalized nature of the outcomes. Especially when first confronted with a new player, be it for balancing between players, or for adjusting the difficulty for an individual player, such DDA or adaptive systems suffer from cold-start problems. Even if the cold-start problems can be overcome and players are provided with a well-matched player experience that results in continued play, adaptive systems suffer the risk of getting stuck in local extrema, or alternatively of enacting strongly fluctuating difficulty settings (rubber banding), which may result in unacceptably balanced play sessions.

Asking players to provide explicit feedback can provide a direction and extent of settings that supports a DDA system to function well. Existing work on user-guided adaptive systems focuses on which information the guidance provides and how it can improve the system [9,50]. User modeling and adaptive systems work also looks at explicit feedback for user-guidance under the term (advisory) dialogue / communication [37] and some approaches are explicitly built around flow theory, with boredom [8] and frustration [27] as critical measures. However, while in traditional software having an explicit dialogue about the adaptive system may often be acceptable, in games, this might endanger immersion [11]. It is therefore important to consider the effects that the presence and presentation of manual difficulty choices have on game user experience.

### Difficulty Choices and Flow in Games

Flow in games has been discussed by a number of researchers in the field [11,13,48]. In this work, we focus on the work by Chen, as the concept of player-oriented (embedded) difficulty choices was introduced in this work [11,12]. He builds on arguments for DDA, suggesting that frequent difficulty adjustments can support flow in games,

and that allowing players to exert user-guidance by providing explicit feedback can provide a feeling of “being in control” and can also inform more adequate adaptations. Chen is cautious that frequent interactions with difficulty setting menus might be disruptive to being immersed in a game, which would in-turn be detrimental to experiencing flow, and so he develops the concept of embedded difficulty choices, which are implemented in such a manner that they blend with the actual game, so that the feeling of being in control as an autonomous actor is provided, while players remain in the magic circle [29] of the game world.

In his line of arguments, Chen relies heavily on flow theory and the theoretical advantages of the embedded difficulty choice approach intuitively appear coherent in this light. He also presents an informal study with two games (called *Traffic Light* and *f10w*) to underline the arguments, with the latter one being specifically designed to implement embedded difficulty choices. However, despite the large number of references to the work in related literature (> 500 references), we could not find an empirical investigation of the effects of embedding player difficulty choices into the core of the interactive experience (or *player-oriented DDA*).

### **Self-Determination Theory and Flow**

Considering the prerequisites for flow, embedded difficulty choices appear prone to be supportive of flow experiences. However, the benefits can also be reasoned based on other motivational theories, such as self-determination theory, which may be beneficial due to the following reasons:

Flow as an indicator for player experiences is highly debated and “*it simplifies the dynamics of intrinsic motivation*” [17] (p. 83), whereas the self-determination theory (SDT) approach has seen growing adoption in player experience research [6,7,46]. SDT in games is assessed using the *Player Experience of Needs Satisfaction* (PENS) questionnaire [42], and results in subscales that can directly inform game design decisions as opposed to the common flow scales [36,49]; flow is a multi-dimensional construct and a matter of present experience – a process variable – that is difficult to measure [5]. For example, the levels of satisfaction of autonomy or competence have been shown to be good indicators of the motivational power of games [40], and they are linked to “feeling in control”. Links between flow and SDT and its measures for intrinsic motivation have been discussed in related work [42] making connections via the aspect of presence / immersion. Csikszentmihalyi also acknowledges similarities between flow and SDT, highlighting the aspect of autonomy. In his view, flow theory arose from an interest in what propels people to initiate or continue actions because they enjoy the performance in the present [17], while other theories (such as needs satisfaction) are more outcome oriented. If SDT and flow theory set a different emphasis (in which SDT is concerned with the preconditions and building factors for intrinsic motivation, and flow theory is concerned with the current and sustained experience of intrinsically motivated

activities), then arguably the major SDT dimensions of competence and autonomy needs satisfaction can be interpreted as provisions for flow experiences, mapping in particular to balance of challenge and skills and the sense of potential and control.

### **EXPECTED IMPACT OF DIFFICULTY CHOICE MODES**

Based on this theoretical background, we explore approaches to game difficulty adjustment in a comparative study with three conditions and the following expectations:

*Menu* – Players select one option from multiple levels of a single difficulty parameter that is presented with through a classic WIMP (windows, icons, menus, pointer) interface.

$H_{mA1}$ : Players will experience higher levels of *autonomy* relative to conditions without explicit choices (here: *auto*).

$H_{mB1}$ : Players will experience reduced *presence / immersion* relative to conditions where the gaming experience is not interrupted by elements that do not blend seamlessly with the game world (here: *embedded* and *auto*).

*Auto* – An implementation of dynamic difficulty adjustments that are performed automatically and where the players are not able to make explicit difficulty choices.

$H_{aA1}$ : Players will experience reduced levels of *autonomy* relative to conditions with explicit difficulty choices.

$H_{aB1}$ : Players will experience increased *presence / immersion* relative to conditions where the gaming experience is interrupted by elements that do not blend seamlessly with the game world (here: *menu*).

*Embedded* – Building on the approach of player-oriented difficulty choices [11], players make explicit difficulty choices by interacting with the game world, integrating closely with the visual design and game mechanics.

$H_{eA1}$ : Players will experience increased levels of *autonomy* relative to conditions where explicit difficulty choices are not possible (here: *auto*).

$H_{eB1}$ : Players will experience increased *presence / immersion* relative to conditions where the gaming experience is interrupted by elements that do not blend seamlessly with the game world.

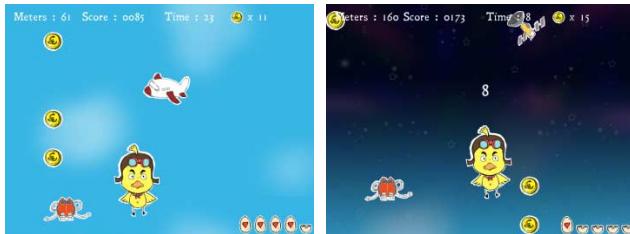
These assumptions were considered most reasonable based on related work, while other effects were also deemed possible. In the *auto* condition, for example, with a DDA system in place, control over difficulty is executed by the system. This may be appreciated by the players, if they are not interested in executing control over this aspect of their interactive experience and the system does not make obvious mistakes. Yet, players might also (to an extent) appreciate full control over those aspects of the system [45]. Conditions with manual (explicit) control might also have an impact on perceived competence need satisfaction; although, in this case, the direction does not appear clear. The impact might be positive (due to perceived control of the system), or negative (e.g. due to becoming aware of requiring “easier” settings). We therefore opted to include

competence in our measures but did not attach a directed hypothesis during our investigation.

We were motivated to this research by our ongoing efforts in the area of motion-based games for health. With applications in the context of therapy (where extrinsic motivation and heteronomy play an important role) regaining competence and autonomy (thus boosting intrinsic motivation) can be valuable. On the other hand, matching the individual capabilities and needs of different members of very heterogeneous target groups is especially important in such use cases and could benefit from the timely adjustment of multiple parameters, calling for automated support with difficulty choices. In this light, the concept of embedded feedback bears the promise that expression takes place in an unobtrusive manner, as a meaningful part of the interactions with the game world.

#### **STUDY 1: THE IMPACT OF DIFFERENT INTERACTION STYLES FOR GAME DIFFICULTY CHOICES**

In order to explore the impact of different modalities for game difficulty choices on the player experience, we conducted a study based on a casual skill game called *The Higher You Fly, The Harder You Fall* (THYFTHYF). In order to attain connectivity of the findings to our ongoing research on full-body motion-based games, the study was implemented as a mixed design with a two level between-groups independent variable of *controller type* being either *motion-based input*, or *gamepad input* and a three level within-subjects independent variable *game difficulty choice interaction modality*. For the purpose of this paper we focus on the within-subjects analysis of the *gamepad* group and only remark on selected between group comparisons.



**Figure 1.** Two screenshots of the game THYFTHYF taken at different height (progress) levels show common game objects.

#### **Design & Implementation**

The game was selected to fulfill a number of requirements: It should transparently and immediately reflect changes in player performance, in order to assure that players would experience effects of their difficulty choices. This includes offering dense in-game and post-game audio-visual feedback. The game was also balanced following pilot study runs to include the possibilities of feeling overstrained, or of losing, even within a short time.

In the game THYFTHYF, the player controls a bird player character (PC) with the goal to fly as high as possible whilst collecting points on the way (cf. Figure 1). It can be played with two types of controls: *motion-based control*, where the player is tracked with a Kinect (v1) and has to move her/his

arms, mimicking the motion with which a bird flaps its wings and *gamepad control* (here: XBOX360 controller), where the player repeatedly pushes and releases the continuous trigger buttons, also mimicking the motion with which a bird flaps its wings. The game world is composed of tiles. Each tile belongs to one of three classes: ground, sky and space. At runtime, the tiles are procedurally placed and populated with “good items”, which increase the player’s score if they are collected by hitting them with the player character and “enemies”, which hurt the player character when contacted and give a sideways impact to the player character that reduces the current balance. The PC can be moved upward by “flapping” both wings at equal speed, where the frequency controls the speed. The direction of flight can be controlled by executing wing-flapping movements with a relatively faster speed on one side (causing the bird to “lean” to the opposite side). If the PC leans too far to either side or stops flapping the wings for too long, it falls down a bit and the player loses a life (starting with a supply of five per round). If all lives are lost, the bird falls all the way to the ground and the player has to start flying up again from ground level with newly refreshed lives. These design decisions were made to assure comparable stimuli; the duration was set to 60 seconds per round. The difficulty, realized by changing speed, balance support and of bad/good objects, increases with increasing height in the level as offsets of base parameters that were influenced by the manual or automatic difficulty choices. After each round, a summary screen showed the final score.

#### *Difficulty Choice Interaction Modalities*

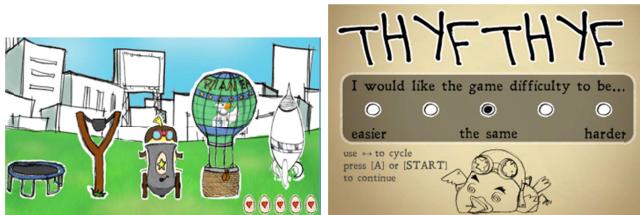
A player-oriented embedded difficulty choice mode was implemented in the form of five different start boosts that reflected the five levels present in the alternative difficulty selection menu (cf. Figure 2). Selections in the difficulty menu would be reflected in a different duration of a jetpack starting boost. Higher boosts would result in increased difficulty settings, whereas lower boosts result in lower difficulty settings. Both modalities were invoked at the beginning of each round in order to assure comparable exposure. The player would also receive a boost start in the DDA only condition without any explicit difficulty choices.

#### *Dynamic Difficulty Adjustments*

Performance based DDA was implemented to allow for a condition without difficulty choices. The mechanism was designed to be limited to a controllable number of effectors, while allowing for distinctly notable adjustments. In the context of the three challenge mechanics that were present in the game (height, collection/avoidance, and balance), the *performance metrics* were defined as follows:

Over the last 5 rounds – weighted by recency –, the height reached, the ratio of bad objects hit, the ratio of good objects collected and the times out of balance were each evaluated by threshold-based heuristics. The balance was set in a way that required the players to perform at a challenging level in order to reach the space level, which always started at a fixed height relative to the final starting

position (after a starting boost), and to assure that players don't struggle with "out-of-balance" events too much, enabling them to collect coins, while avoiding enemies. Before each session, the according *difficulty parameters* of max speed / fall speed, number of bad / good objects, and balance support (via damping) were adjusted.



**Figure 2: Left/right: Embedded difficulty choice / menu difficulty choice in THYFTHYF.**

THYFTHYF was implemented in C# based on .NET 3.5, the XNA 4.0 framework and the Kinect 1.8 SDK.

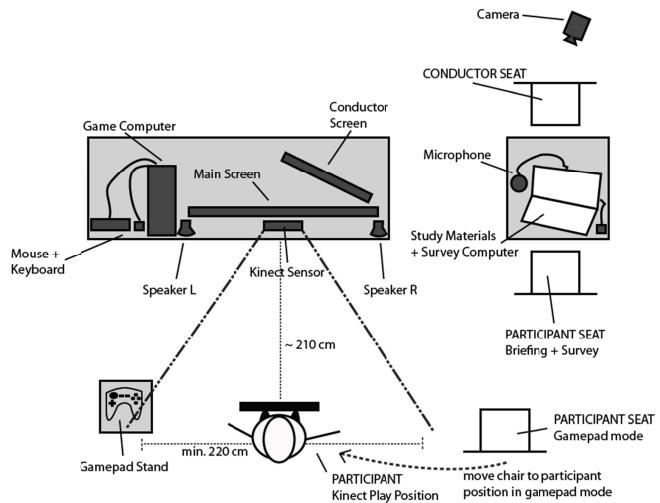
### Setup & Procedure

Trials of the laboratory study lasted about 60 minutes. We recruited from the local student body, and each participant received \$10 CAD compensation for their time. Figure 3 illustrates the technical setup. The study language was English. The procedure was performed with 40 participants (19f, 21m, age: M = 26.88, SD = 5.79, min/max = 18/45), who were randomly assigned to the sedentary (gamepad) or the motion-based (Kinect) group. Four participants said they never play video games in a typical week, 16 said they play up to three hours per week and two participants said they play more than 35 hours/week, with the rest spending between 3 hours and 35 hours a week gaming. They have been gaming since 10.76 years on average (SD = 6.7).

Following the greeting and gathering of informed consent, participants were asked to complete a pre-study questionnaire before engaging in a warmup round with a short introduction by demonstration and explanation of the core game mechanics. Each subject then participated in four trials of gameplay consisting of three one-minute rounds interspersed with a difficulty choice in the conditions *embedded* and *menu*, followed by a fixed set of post-trial questionnaires. The fourth (and last) trial always featured a zero effect menu choice placebo condition that was omitted from analysis and report for brevity. The difficulty choice modality conditions *embedded*, *menu*, and *none* were presented in Latin square randomized order to counter-balance potential learning-, customization- and fatigue-effects. Lastly, the participants completed a short personality trait index and responded to a semi-structured interview about experiences and preferences.

The independent variable of *controller type* (motion-based vs. sedentary) was added to the design in order to facilitate carrying over results, because potential interaction effects (e.g. of feeling more / less in control or autonomous when exposed in regard to one's physical appearance and skills while playing motion-based games) might exist. We found

no evidence for meaningful interaction effects and for reasons of brevity, the analysis of that group was largely omitted from this report. Notable fixed variables include design choices around interacting with explicit user feedback, which was always collected between one round of gameplay and the next (fixed intervals) and not based on the players' volition to assure comparable outcomes. Players were informed that they could influence the difficulty of the game and would be asked to do so between rounds. It was explained to them, what each difficulty choice in both the *menu* and the *embedded* condition would mean before the respective conditions.



**Figure 3: The technical setup of the study.**

### Measures

In addition to demographic background information, gaming preferences, and a brief personality questionnaire (TIPI), we measured *affect* with the *Positive and Negative Affect Schedule* (PANAS) [15], *player experience* with the *Player Experience of Needs Satisfaction* (PENS) [39] questionnaire (the dimensions *interest-enjoyment* and *effort-importance* of the *Intrinsic Motivation Inventory* (IMI) [35] were added as additional dimensions rooted in SDT to augment the PENS results), and a *Task Load Index* (TLX) was collected as a measure of *frustration* rooted in usability research, as the interaction with difficulty choices was realized either "inside" a game, or with regular GUI components. The questionnaires were presented in the order of introduction in this text and with randomized item order. Each questionnaire was completed multiple times by each participant (once after each trial) and the PANAS was also completed once before the first trial in order to facilitate relative offset calculations that compensate for individual differences in initial affect. The participants were instructed to consider their entire interaction sessions with the game (including interface components). Before the final interview, players also reported a ranking of their preference for the difficulty choice modalities together with responses to a number of question items regarding their usual perception of – and interaction with – difficulty

choices in games. Video recordings and observational protocols completed the data collection.

### Results & Analysis

The quantitative experiential measures were analyzed with a general linear model facilitating a mixed design repeated-measures analysis of variance (ANOVA) at a significance level of  $\alpha = .05$  and Mauchly's sphericity tests, as well as Bonferroni adjustments prior to post-hoc pairwise t-tests for multiple comparisons. The analysis was performed in R with the *ezStats* package for the ANOVA operations, and variances were winsorized [19] (leveling outliers in the top/bottom .2 quantiles to the trim edge values). The results were confirmed using SPSS (version 20) and are summarized in Table 1 and Table 2.

| STUDY 1<br>(PANAS, PENS, IMI) | <b>embedded</b><br>[M (SD)]: | <b>menu</b><br>[M (SD)]: | <b>auto</b><br>[M (SD)]: |
|-------------------------------|------------------------------|--------------------------|--------------------------|
| <b>positive affect</b>        | 2.95 (.77)                   | 2.96 (.68)               | 2.94 (.82)               |
| <b>positive affect (MB)</b>   | 3.37 (.74)                   | 3.44 (.75)               | 3.29 (.74)               |
| <b>negative affect</b>        | 1.59 (.53)                   | 1.58 (.64)               | 1.54 (.46)               |
| <b>competence</b>             | 2.72 (.9)                    | 2.8 (.7)                 | 2.83 (.81)               |
| <b>presence / immersion</b>   | 2.83 (.74)                   | 2.84 (.73)               | 2.73 (.71)               |
| <b>autonomy</b>               | 3.07 (.75)**                 | 2.9 (.74)                | 2.68 (.9)**              |
| <b>relatedness</b>            | 2.73 (.68)                   | 2.77 (.75)               | 2.65 (.97)               |
| <b>intuitive control*</b>     | 2.93 (.72)                   | 3.03 (.74)               | 3.11 (.88)               |
| <b>intuitive cont. (MB)*</b>  | 3.52 (.76)                   | 3.57 (.55)               | 3.57 (.77)               |
| <b>interest enjoyment</b>     | 3.34 (.53)                   | 3.38 (.47)               | 3.3 (.61)                |
| <b>effort-importance</b>      | 3.71 (.61)                   | 3.71 (.44)               | 3.7 (.62)                |

**Table 1: Mean (M) and standard deviation (SD) results of study 1. All items were recorded on 5 pt. Likert scales. Group effects comparing between sedentary and motion-based (MB) are indicated with (\*). Within-subjects effects between the embedded, menu, and auto conditions are indicated with (\*\*).**

There were no significant effects on the *positive affect* or on the *negative affect* scale. Notably higher *positive affect* than *negative affect* and also higher *positive affect* in the *motion-based* (MB) game group are in line with the following measures and suggest that the scale is sensitive to changes in affect caused by the experience of playing different versions of the game. While the lack of significances does not prove the absence of effects, we assume that there are no large effects on *affect* between the three conditions, because the means between all conditions are very close.

The PENS results on the *competence* and *presence / immersion* dimensions show a similar picture. While the result on *competence* appeared difficult to predict due to the complex interaction of self-perceived, practical skill and interacting with difficulty choices, *presence / immersion* could be expected to be lowered in the *menu* condition, which was not the case in our sample and the similarity in means suggests an absence of strong effects. There was a significant difference on the PENS *autonomy* dimension ( $F(2,76) = 5.01$ ,  $p = .009$ , gen.  $\eta^2 = .02$  [4], Mauchly not sig.) with post-hoc pairwise comparisons confirming a sig. diff. between *embedded* and *auto* ( $p = .02$ ). This finding can be seen as evidence to support an increased sense of autonomy needs satisfaction in the *embedded* condition that was predicted based on Chen's arguments. However, there

is no discernable difference between the *embedded* and the *menu* condition. The PENS dimensions *relatedness* and *intuitive control* showed no significant differences on the indep. variable *difficulty selection*, as expected, while *intuitive control* was sig. increased under the *motion-based* control condition ( $F(1,38) = 6.33$ ,  $p = .016$ , gen.  $\eta^2 = .12$  [4], Mauchly not sig.), which could also be expected, as the game was originally designed to be motion-based.

The IMI dimensions *interest-enjoyment* and *effort-importance* showed remarkable similarity in means and no significant differences, suggesting that both player enjoyment and motivation were not notably different under the different difficulty choice modalities.

| STUDY 1<br>(TLX)        | <b>embedded</b><br>[M (SD)]: | <b>menu</b><br>[M (SD)]: | <b>auto</b><br>[M (SD)]: |
|-------------------------|------------------------------|--------------------------|--------------------------|
| <b>physical demand*</b> | .8 (10.33)                   | .65 (10.93)              | 1.15 (10.68)             |
| <b>phys. dem. (MB)*</b> | 11.05 (4.76)                 | 8.35 (6.18)              | 9.5 (6.41)               |
| <b>mental demand</b>    | 2.9 (7.02)                   | 3.15 (7.34)              | 3.1 (8.5)                |
| <b>temporal demand*</b> | 7.35 (5.37)                  | 7.1 (3.87)               | 6.75 (3.35)              |
| <b>temp. dem. (MB)*</b> | 2.3 (8.02)                   | 2.35 (8.7)               | 2.05 (10.45)             |
| <b>performance*</b>     | -1.9 (7.72)                  | -3 (8.3)                 | -1 (9.46)                |
| <b>perf. (MB)*</b>      | 3.65 (8.42)                  | 4.15 (7.21)              | 3.45 (7.07)              |
| <b>effort</b>           | 8.15 (3.82)                  | 8.25 (3.6)               | 7.4 (5.06)               |
| <b>frustration</b>      | -.75 (8.4)                   | -1.65 (8.93)             | -1.95 (9.78)             |

**Table 2: The mean (M) and standard deviation (SD) results of the TLX in study 1. All items were recorded on 40 pt. Likert scales (range -20 to 20). Group effects comparing between sedentary and motion-based (MB) are indicated with (\*).**

The TLX dimension *physical demand* showed no differences between the within-subject conditions, although the *motion-based* game group recorded significantly higher *physical demand* ( $F(1,38) = 11.6$ ,  $p = .002$ , gen.  $\eta^2 = .22$  [4], Mauchly not sig.), providing further evidence for the sensibility of the chosen measures. The TLX *mental demand* and *temporal demand* dimensions showed no significant differences, although the mean of the *temp. dem.* *auto* condition appears lowered, which seems reasonable given the lack of a manual selection process. *Temporal demand* appears sig. decreased in the *motion-based* game group ( $F(1,38) = 5.66$ ,  $p = .023$ , gen.  $\eta^2 = .11$  [4], Mauchly not sig.), which cannot be explained by observable differences in actual time spent and seems contradictory to the physical effort measure, hinting at a potential interaction with overall motivational effects. There were no sig. diffs. on *difficulty choice* in the TLX *performance* dimension, although there was a sig. diff. between *sedentary* and *motion-based* ( $F(1,38) = 4.43$ ,  $p = .042$ , gen.  $\eta^2 = .09$  [4], Mauchly not sig.), which may be a secondary effect to the observed difference in *physical demand*. While THYFTHYF appears to be a rather high-effort game, the final TLX dimensions of *effort* and *frustration* showed no further sig. diffs., hinting at a further lack of notable negative or positive effects of the *difficulty choice modality*.

### Interviews

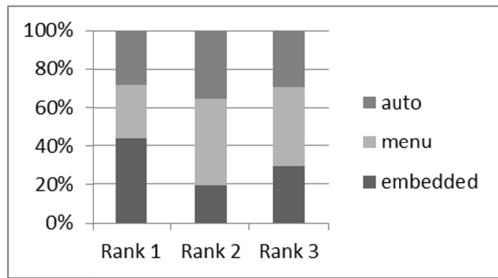
In the interviews, three out of four participants said that they *like being able to change settings*, as opposed to being *more happy just playing the game*. Regarding the difficulty

selection in THYFTHYF, one participant expressed it this way "*I did like being able to choose the difficulty settings because it felt like I could cater the game to how well I played, but I also liked playing the game itself*", while some others expressed notions such as: "*I'm just more happy playing the game*". Some participants added that they *appreciate a broad range of (difficulty) settings*.

When asked *how they decided which difficulty to select*, participants mentioned various strategies they followed to make difficulty choices ranging from *depending on last time*, over *first easier then harder and just stay in the middle*, to *just hard and first easiest, then hardest*, and in three cases even just *random* (which was only mentioned with embedded difficulty selection but not regarding the menu). A common notion was "*I just did it based on how it felt the previous time I played it*".

Players expressed a broad range of opinions regarding how difficult they found it to play the game, ranging from *too easy* (2) over *okay / all right* to *too hard* (4 mentions, 3 of those relating to difficult controls; occurring in both controller type groups). 20 participants added that they generally *prefer games to be challenging (or hard)*.

Regarding whether they felt that their difficulty selections made a difference, 22 participants explicitly said they *noted that their difficulty selections made a difference*. Three did not clearly reply to this question, while 15 did *not clearly say that they noted differences*. However, some added positive unasked remarks on the DDA, such as "*I liked the idea that the game difficulty was automatically adjusted to my performance*". In those parts of the open-ended discussion elements that led to players making a statement whether they like the idea of DDA in general, 9 were positive about DDA, while 6 expressed rather negative connotations. During the post-study interviews, the participants were eventually informed about the difficulty selection interaction modality manipulation and later question items named the conditions, which may explain why some participants expressed preferences for automatic adjustments or manual adjustments.



**Figure 4: Ranking by participants regarding preference in study 1 (with the game THYFTHYF).**

The ranking of all conditions (cf. Figure 4) in comparison after completion of the trials does not show significant results on the Friedman rank sum test. While *embedded* received slightly more first rank preferences, Exact

Wilcoxon-Pratt Signed-Rank Tests also did not show any notable contrasts in pairwise comparisons.

While the findings seem to support Chen's cautioning of an impact on the player *autonomy, presence/immersion* does not appear to be impacted and the difficulty choice modality similarly has *no notable impacts* on a large variety of enjoyment related measures. We were intrigued by the lack of significant effects on presence / immersion and any secondary or related experiential measures (especially interest-enjoyment, effort-importance, and affect).

## STUDY 2: THE IMPACT OF DIFFERENT LEVELS OF FEEDBACK EXPLICITNESS IN CHEN'S GAME FLOW

Results from study 1 showed a surprising homogeneity, especially of the resulting directly enjoyment-related measures, which were thought to react to the different game difficulty choice modes. This partially contradicted predictions that we drew from Chen's considerations. However, Chen's arguments relate directly to his own game implementations. Our game THYFTHYF – while also being a casual game – bears quite a number of differences, especially with regard to how difficulty choices were triggered and how the embedded difficulty choice mode integrated into the game world and the game mechanics. We thus repeated the study in a comparable setup with Chen's well-known game *f1ow* in order to further investigate the unexpected absence of effects and to seek confirmation for the effect on autonomy.

### Design & Implementation

In the game *f1ow*, which can be described as a casual atmospheric game, the player controls a microorganism that can chase and "eat" (collide with the mouth section) available food items that float around in multiple fluid-like layers of a game world that may have been imagined as a petri dish. The player organism moves towards the position of the mouse cursor on screen, when the left button is clicked. With increasing "depth", other microorganisms are present in the layers and some will attack the player's organism by attempting to "eat" some of its components. The player organism can similarly "fight" other organisms by "eating them" piece-by-piece. The embedded difficulty choice mode originally implemented in the game allows players to move up and down through different difficulty layers of their own volition by eating either a special red piece (one layer down; harder) or a special blue piece (one layer up; easier) as shown in the instructions screenshot in Figure 5. The game *f1ow* was converted to *Haxe* based on the *AS2* source code made available by J. Chen [10].

### Difficulty Choice Interaction Modalities

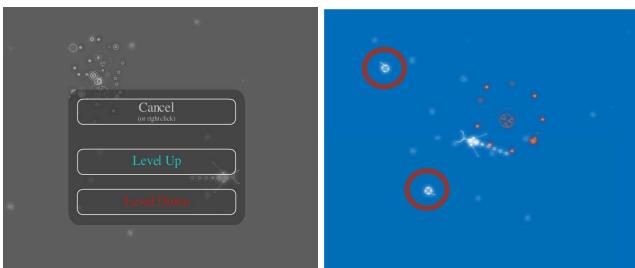
In addition to the original embedded difficulty choices method, we implemented a traditional menu, which could be opened at any time during play by clicking the right mouse button (see Figure 6), as well as a fully automatic DDA method, relating to the methods *embedded*, *menu*, and *auto* from study 1. The most notable difference in difficulty selection, aside from the selection upon the players'

volution at arbitrary times during game play in *f10w* was that THYFTHYF uses a five point difficulty choice scale, whereas the difficulty choice in *f10w* is binary.

Eat this food ☺ to go one level down.  
And eat this food ☻ to go one level up.

**Figure 5: Instructions for the condition *embedded*.**

Instructions provided for the condition *menu* were: “Click the right mouse button at any time to change the level.”



**Figure 6: Difficulty choice interaction styles for the condition *menu* (left) and *embedded* (right; red circles are clarification markers not present in the game). Left background elements and right game assets: © Jenova Chen / thatgamecompany.**

#### Dynamic Difficulty Adjustments

In order to automatically increase difficulty, a timer-based heuristic check was performed regularly to determine whether any other organisms or food items were still visible to the player. If there were no further other organisms alive in the layer, as well as no more than two food items available, the difficulty level was increased by moving one layer down. The mechanism for automatically decreasing difficulty was kept as originally implemented in *f10w*, meaning that the player’s creature would automatically be moved up one layer if its health status (as expressed by intact body segments) fell one level due to an attack by another creature.

#### Setup & Procedure

The study setup duplicated the setup from study 1 with the following adjustments for brevity: The BPNS and TIPI personality trait questionnaires were skipped, as they were not of interest to this comparison study. Study 2 also did not feature interviews or a motion-based game condition (safe to omit due to the between groups mixed design of study 1). Latin square randomization was maintained for the order of within-subjects conditions.

The study was completed by 18 participants (7f, 11m, age: M = 26.78, SD = 5.71, min/max = 20/45) to test whether effects with notable effect sizes would occur. Four participants said they never play video games in a typical week, seven said they play up to 3 hours per week and none said that they played more than 21 hours per week. They have been gaming since 13.28 years on average (SD = 6.64). Participants did not receive monetary compensation and were conveniently sampled mostly from a student population in Germany. The study language was English.

#### Results & Analysis

There were no significant effects on the *positive affect* or on the *negative affect* scale (cf. Table 3), mirroring results of study 1 and suggesting again the absence of large effects between the three conditions, because the means are very similar. Also in agreement with study 1, the PENS results on the *competence* and *presence / immersion* dimensions show a similar picture with the lack of an effect on *presence / immersion* in contrast to the *menu* condition, contradicting the expectations. The significant difference on the PENS *autonomy* dimension from study 1 was confirmed ( $F(2,34) = 4.17$ ,  $p = .024$ , gen.  $\eta^2 = .05$  [4], Mauchly not sig.) with post-hoc pairwise comparisons confirming a sig. diff. between *menu* and *auto* ( $p = .047$ ). This finding can be seen as evidence to support an increased sense of *autonomy* needs satisfaction in the *menu* condition that was predicted based on the Chen’s arguments, providing a helpful complement to the significant difference found in post-hoc pairwise comparison between the *embedded* and *auto* condition in study 1, while again finding very similar means in the *embedded* and the *menu* conditions. The PENS dimensions *relatedness* and *intuitive control* showed no significant differences, as expected, and in agreement with study 1. Findings were also repeated for the IMI dimensions *interest-enjoyment* and *effort-importance*, which show remarkable similarity in means, including the *embedded* condition, and no significant differences.

| STUDY 2<br>(PANAS, PENS, IMI) | embedded<br>[M (SD)]: | menu<br>[M (SD)]: | auto<br>[M (SD)]: |
|-------------------------------|-----------------------|-------------------|-------------------|
| <b>positive affect</b>        | 26.79 (3.57)          | 26.97 (5.65)      | 26.02 (5.72)      |
| <b>negative affect</b>        | 12.57 (2.65)          | 12.13 (1.83)      | 12.94 (1.99)      |
| <b>competence</b>             | 3.37 (.6)             | 3.47 (.49)        | 3.61 (.47)        |
| <b>presence / immersion</b>   | 2.67 (.62)            | 2.6 (.77)         | 2.63 (.7)         |
| <b>autonomy</b>               | 3.06 (.7)             | 3.17 (.67)*       | 2.84 (.47)*       |
| <b>relatedness</b>            | 1.69 (.66)            | 1.72 (.6)         | 1.65 (.6)         |
| <b>intuitive control</b>      | 4.42 (.51)            | 4.35 (.39)        | 4.56 (.45)        |
| <b>interest enjoyment</b>     | 3.67 (.5)             | 3.82 (.36)        | 3.64 (.39)        |
| <b>effort-importance</b>      | 2.89 (.67)            | 2.69 (.58)        | 2.71 (.39)        |

**Table 3: Mean (M) and standard deviation (SD) results of study 2. All items were recorded on 5 pt. Likert scales (pos. and neg. affect use mean sums of item scores; range 10 - 50). Within-subjects effects are indicated with (\*).**

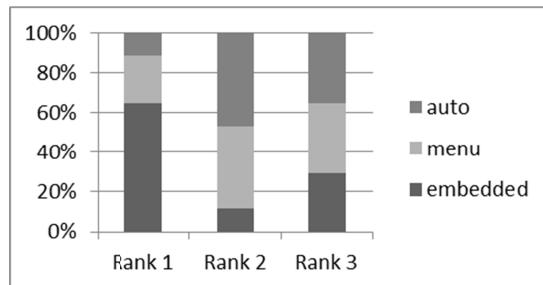
The TLX dimensions *physical demand* and *performance* (cf. Table 4) showed no significant differences, also repeating the findings from study 1. The TLX *effort* ( $F(2,34) = 6.47$ ,  $p = .004$ , gen.  $\eta^2 = .11$ , Mauchly not sig.) and *frustration* ( $F(2,34) = 3.6$ ,  $p = .038$ , gen.  $\eta^2 = .07$ , Mauchly not sig.) dimensions did, however, show sig. diff., with post-hoc pairwise comparisons showing a sig. diff. between *embedded* and *manual* on the *effort* dimension ( $p = .011$ ) and between *embedded* and *auto* on the *frustration* dimension ( $p = .047$ ). This finding is not in line with the results from study 1, yet it appears reasonable considering the game design of *f10w*, as players are required to make an effort to chase the special food item of their liking in order to move one layer up or down (which affects the difficulty level accordingly). That argument can be supported with the results from the *mental demand*, and the

*temporal demand* dimensions, which show higher mean scores for the *embedded* condition and for which the ANOVA returns a trend ( $F(2,34) = 3.24$ ,  $p = .051$ , gen.  $\eta^2 = .04$ , Mauchly not sig.) in the case of *temporal demand*. Pairwise comparison hints at the contrast between *embedded* and *auto* ( $p = .069$ ) as the main cause.

| STUDY 2<br>(TLX)       | embedded<br>[M (SD)]: | menu<br>[M (SD)]: | auto<br>[M (SD)]: |
|------------------------|-----------------------|-------------------|-------------------|
| <b>physical demand</b> | 2.36 (1.03)           | 2.7 (1.07)        | 2.72 (.89)        |
| <b>performance</b>     | 13.87 (3.45)          | 13.81 (3.15)      | 14.03 (2.77)      |
| <b>effort</b>          | 8.28 (2.32)*          | 5.97 (2.36)*      | 6.76 (2.69)       |
| <b>frustration</b>     | 4.52 (2.25)*          | 3.8 (1.86)        | 3.18 (1.94)*      |
| <b>mental demand</b>   | 8.52 (3.82)           | 7.91 (4.55)       | 7.56 (4.72)       |
| <b>temporal demand</b> | 8.67 (3.36)           | 7.37 (3.58)       | 7.17 (3.34)       |

**Table 4: The mean (M) and standard deviation (SD) results of the TLX in study 2. All items were recorded on 20 pt. Likert scales (range 1 to 20). Sig. effects are indicated with (\*).**

The ranking of all conditions (cf. Figure 7) in comparison after completion of the trials does not show significant results on the Friedman Rank Sum Test ( $\chi^2(2) = 3.44$ ,  $p = .18$ ), although *embedded* received the highest total score and an Exact Wilcoxon-Pratt Signed-Rank Test comparing *embedded* with *auto* shows a trend ( $Z = 1.82$ ,  $p = .08$ ,  $r = .3$ ; suggesting a medium effect size).



**Figure 7: Ranking by participants regarding preference in study 2 (with the game *flow*).**

Study 2 repeated the specific player experience items from study 1 almost exactly. We confirmed an effect on *autonomy*. There are significant differences between study 1 and 2 in the TLX dimensions *effort* and *frustration*, which are supported by differences in means on the *mental demand* and *temporal demand* dimensions. If interpreted as a classical “task load index”, these measures put *embedded* in a worst position regarding usability. For game design choices, high task loads arguably need not necessarily be seen as adverse, because – given the right circumstances – overcoming challenges in a game generates positive player experiences (the similarity in most PENS and IMI player experience scales may be interpreted to support this argument). At the same time, frustrating controls can hinder game experiences. Either way, game designers can benefit from an awareness of possible frustrations triggered by their game difficulty choice implementations.

## DISCUSSION

In this paper, we presented two studies on the impact of game difficulty choice interaction modes on player

experience, focusing on potential effects on autonomy and presence/immersion for which outcome expectations were discussed based on related work.

We found evidence for a reduced autonomy needs satisfaction if no explicit difficulty choice is present, which appears in line with the expectations given the presence of manual difficulty choices in the other conditions. However, we could neither find a significant impact on presence / immersion in the case of non-embedded difficulty choices nor on any other player experience measure (with the reasonable exception of TLX measures in study 2), indicating that the differences in perceived autonomy did not have any strong impact on the overall enjoyment, motivation and affect; or in short on most of the game experience. The significant differences and trends measured on the TLX dimensions mentioned in the analysis of study 2 point towards the embedded difficulty choice being a potential source of frustration, due to the considerable effort that it required when seeking out the special food items that are not always in the same position and not always in close reach, thereby breaking interaction best practices when considered from a usability point of view. While further investigations about how players perceive difficulty choices in the forms of game mechanical elements and settings interfaces could help clarify such assumptions, the notable closeness of all modalities on most player experience measures indicates that the impact of the game difficulty selection modality on the overall game experience is likely to be small enough to allow for other considerations to guide game designers’ choices of modality. With complex motion-based games for health, for example, where large numbers of sensible difficulty-related variables can be influenced by difficulty adjustments, this suggests that a strong role of DDA / adaptivity could lead to player experiences that are at least comparable to those that can be achieved when notable manual difficulty choices are present, even if they were well embedded.

While elements of choice have been linked to increased cognitive and affective engagement in learning scenarios [20], the suggested link to game enjoyment may not have results that are as large as an intuitive theoretical understanding suggests. The strong homogeneity in results between studies 1 and 2 suggests that these negative results were not accidental for the genre of casual games, and the similarities in means across conditions in the measures without significant findings indicate that strong effects on the overall player experience are unlikely to exist. The comparison with the motion-based group in study 1 serves as an indicator that the employed measures did pick up on relevant player experience factors, adding to our confidence in interpreting the large number of homogenous results.

The TLX as a classic demand measure was included because it has been used in the context of games by other researchers [24,38]. Our study provides further evidence for the sensitivity of the separate dimensions in game user

research. While it requires careful interpretation in games, as desirable challenges may be part of the game design, it can be seen as an asset in the young area of game user research, where reliable psychometrics are still rare.

### Limitations & Future Work

A number of limitations should be noted. Explicit feedback to an adaptive system is an unusual game design element. A further study that investigates the framing of “pre-play settings” (e.g. settings menu) vs. “post-play feedback” might thus deliver interesting additional insights. However, from our observation and the user responses during the interviews, we gather that the menu-feedback condition was seen as being similar to typical difficulty settings that are usually accessible through a main menu. It is also not clear in how far experiences related to flow are triggered in short episodes of casual game-play, although other researchers suggest that flow can emerge from a broad range of durations of captivating activities [11,17]. More highly immersive, longer-term gaming sessions might lead to different reactions to the offered difficulty choice modes.

A number of specific game design choices that were made when preparing the games and setup offer opportunities for further investigations in alternative design choices. The point of intervention for the DDA, for example, was set to be between rounds for study 1, whereas continuous or sparser adjustments are also feasible. Likewise, if an adaptive mechanism is present, feedback can strongly influence the DDA settings, adjusting not only the specific game mechanical variables, but also the estimated optimal performance thresholds for an individual player (moving from fixed threshold DDA to an individual solution). Other feedback mechanisms, such as (implicit) general affective feedback, more specific technical feedback, or feedback on multiple dimensions, may be taken into account, and other intervals for providing feedback (or other modalities to express volition) could be evaluated. Because both games were “casual”, many other game types offer opportunities for further investigations on game difficulty choices. Due to the differences between the embedded difficulty choice modalities in study 1 and 2 (i.e., it was always available yet required effort to attain in study 2, whereas it was only available once per game round in study 1), further studies might expand on our results by focusing on frequency and ease of access as independent variables.

### CONCLUSION

We presented an investigation of the impact of difficulty adjustments that were performed either through manual control via a classic selection menu, through an embedded difficulty choice, or with an automatic heuristics-based DDA system, on game user experience. With prior considerations regarding the impact based largely on flow theory, we presented a transfer of the expectations to the dimensions measured by validated game user research tools rooted in self-determination theory. The expected effect on autonomy was observed in the data based on a study with a

game of our own design, however, no impact on presence / immersion could be found, which contradicted our outcome expectations. In the interviews, many players expressed that they prefer the presence of manual difficulty choices in games, yet our participants were more likely to be positive about DDA than to be negative about it and did not remark negatively on the presence of DDA. This ambiguity, together with a lack of notable differences on any resulting game experience measure besides the PENS dimension of autonomy, prompted us to repeat the study design with the game *f10w*. Because Chen employed the latter to argue for the benefits of his concept of embedded (player-oriented) difficulty choices, we aimed to double-check for potential biases introduced with specific game design decisions of our game. The findings were largely repeated, including the surprisingly similar means across conditions on almost all game experience dimensions. Due to the repeated absence of effects in both studies, we find the evidence to be worth reporting. Significant differences in the TLX dimensions of effort and frustration in study 2 were found; these can be explained by game design aspects of the *embedded* mode in *f10w*, which requires players to manually seek out special game objects in order to influence the game difficulty. Since no game motivation related measure besides autonomy showed an impact between conditions in either study, we conclude that in practical (casual) game design, all versions could lead to a very similar game experience.

Our findings suggest that the game difficulty choice interaction mode in casual games might play a minor role compared to other game design choices. These could therefore be prioritized by game designers. Letting other considerations influence the choice of game difficulty interaction modality, or opting for common and simple difficulty selection menus, appear to be reasonable choices. In terms of game user research and arguments on game design based in motivation theory, we could only partially confirm our expectations, which appeared firmly rooted in theory and intuitively logical. Hence, we can only support the call for empirical investigations of game design theory so that designers are enabled to make more certain, well-informed, and detailed decisions.

### ACKNOWLEDGMENTS

We thank the GRAND NCE, the Klaus Tschira Stiftung, as well as the Federal Ministry of Education and Research, Germany (BMBF) for funding, and members of the Interaction Lab at the University of Saskatchewan, the Digital Media Lab at the University of Bremen, the original THYFTHYF developer team, as well as all study participants, Rodrigo Vicencio-Moreira, Jenny Cramer, Jenova Chen, and thatgamecompany for their kind support.

### REFERENCES

1. Ernest Adams. 2010. *Fundamentals of Game Design*. New Riders.
2. Gazihan Alankus, Amanda Lazar, Matt May, and Caitlin Kelleher. 2010. Towards customizable games

- for stroke rehabilitation. *Proceedings of the 28th international conference on Human factors in computing systems*, ACM, 2113–2122. <http://doi.org/10.1145/1753326.1753649>
3. Christine Bailey and Michael Katchabaw. 2005. An experimental testbed to enable auto-dynamic difficulty in modern video games. *Proceedings of the 2005 GameOn North America Conference*, 18–22. Retrieved September 21, 2015 from [http://mmm.csd.uwo.ca/faculty/katchab/pubs/gameon2005\\_add.pdf](http://mmm.csd.uwo.ca/faculty/katchab/pubs/gameon2005_add.pdf)
  4. Roger Bakeman. 2005. Recommended effect size statistics for repeated measures designs. *Behavior Research Methods* 37, 3: 379–384. <http://doi.org/10.3758/BF03192707>
  5. Gary Bente. 2009. Making the implicit explicit. Embedded measurement in serious games.
  6. Max Birk and Regan L. Mandryk. 2013. Control your game-self: effects of controller type on enjoyment, motivation, and personality in game. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 685–694. <http://doi.org/10.1145/2470654.2470752>
  7. Max V. Birk, Regan L. Mandryk, Matthew K. Miller, and Kathrin M. Gerling. 2015. How Self-Esteem Shapes Our Interactions with Play Technologies. *Proceedings of the 2015 Annual Symposium on Computer-Human Interaction in Play*, ACM, 35–45. <http://doi.org/10.1145/2793107.2793111>
  8. Guillaume Chanel, Cyril Rebetez, Mireille Bétrancourt, and Thierry Pun. 2008. Boredom, engagement and anxiety as indicators for adaptation to difficulty in games. *Proceedings of the 12th international conference on Entertainment and media in the ubiquitous era*, ACM, 13–17. <http://doi.org/10.1145/1457199.1457203>
  9. Darryl Charles and Michaela Black. 2004. Dynamic player modeling: A framework for player-centered digital games. *Proc. of the International Conference on Computer Games: Artificial Intelligence, Design and Education*, 29–35. Retrieved November 26, 2012 from <http://research.rmutp.ac.th/paper/cu/DynamicPlayerModelling.pdf>
  10. Jenova Chen. 2006. *flow - source code*. Retrieved September 25, 2015 from <http://www.thatgamecompany.com/forum/viewtopic.php?p=4255&sid=5af8dfad64831dfb41b7bc4e66c1a165#p4255>
  11. Jenova Chen. 2007. Flow in games (and everything else). *Commun. ACM* 50, 4: 31–34. <http://doi.org/10.1145/1232743.1232769>
  12. Jenova Chen. Flow in Games. Retrieved September 19, 2011 from [http://jenovachen.com/flowingames/Flow\\_in\\_games\\_final.pdf](http://jenovachen.com/flowingames/Flow_in_games_final.pdf)
  13. Ben Cowley, Darryl Charles, Michaela Black, and Ray Hickey. 2008. Toward an Understanding of Flow in Video Games. *Comput. Entertain.* 6, 2: 20:1–20:27. <http://doi.org/10.1145/1371216.1371223>
  14. Chris Crawford. 1984. *The Art of Computer Game Design*. Osborne/McGraw-Hill.
  15. John R. Crawford and Julie D. Henry. 2004. The Positive and Negative Affect Schedule (PANAS): Construct validity, measurement properties and normative data in a large non-clinical sample. *British Journal of Clinical Psychology* 43, 3: 245–265.
  16. Mihaly Csikszentmihalyi. 1990. *Flow: The Psychology of Optimal Experience*. Harper & Row, New York.
  17. Mihaly Csikszentmihalyi and K. Rathunde. 1992. The measurement of flow in everyday life: toward a theory of emergent motivation. Nebraska Symposium on Motivation. Nebraska Symposium on Motivation 40: 57–97.
  18. Anders Drachen, Alessandro Canossa, and Georgios N. Yannakakis. 2009. Player modeling using self-organization in tomb raider: underworld. *Computational Intelligence and Games*, 2009. CIG 2009. IEEE Symposium on, 1–8.
  19. David M. Erceg-Hurn and Vikki M. Mirosevich. 2008. Modern robust statistical methods: An easy way to maximize the accuracy and power of your research. *American Psychologist* 63, 7: 591–601. <http://doi.org/10.1037/0003-066X.63.7.591>
  20. Terri Flowerday and Gregory Schraw. 2003. Effect of Choice on Cognitive and Affective Engagement. *The Journal of Educational Research* 96, 4: 207–215. <http://doi.org/10.1080/00220670309598810>
  21. Sandro Franceschini, Simone Gori, Milena Ruffino, Simona Viola, Massimo Molteni, and Andrea Facoetti. 2013. Action Video Games Make Dyslexic Children Read Better. *Current Biology*. <http://doi.org/10.1016/j.cub.2013.01.044>
  22. Yue Gao and Regan L. Mandryk. 2012. The Acute Cognitive Benefits of Casual Exergame Play. *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*, 1863–1872.
  23. Kathrin Maria Gerling, Frank Paul Schulte, Jan Smeddinck, and Maic Masuch. 2012. Game Design for Older Adults: Effects of Age-Related Changes on Structural Elements of Digital Games. In *Entertainment Computing - ICEC 2012*, Marc Herrlich, Rainer Malaka and Maic Masuch (eds.).

- Springer Berlin Heidelberg, 235–242. Retrieved January 18, 2013 from [http://link.springer.com/chapter/10.1007/978-3-642-33542-6\\_20](http://link.springer.com/chapter/10.1007/978-3-642-33542-6_20)
24. Kathrin M. Gerling, Kristen K. Dergousoff, and Regan L. Mandryk. 2013. Is Movement Better?: Comparing Sedentary and Motion-based Game Controls for Older Adults. *Proceedings of Graphics Interface 2013*, Canadian Information Processing Society, 133–140. Retrieved September 21, 2015 from <http://dl.acm.org/citation.cfm?id=2532129.2532153>
  25. Kathrin M. Gerling, Regan L. Mandryk, Matthew Miller, Michael R. Kalyn, Max Birk, and Jan D. Smeddinck. 2015. Designing Wheelchair-Based Movement Games. *ACM Trans. Access. Comput.* 6, 2: 6:1–6:23. <http://doi.org/10.1145/2724729>
  26. Kathrin M. Gerling, Matthew Miller, Regan L. Mandryk, Max Birk, and Jan Smeddinck. 2014. Effects of Balancing for Physical Abilities on Player Performance, Experience and Self-Esteem in Exergames. *CHI'14: Proceedings of the 2014 CHI Conference on Human Factors in Computing Systems*. Retrieved February 11, 2014 from <https://hci.usask.ca/uploads/331-paper122.pdf>
  27. Kiel M. Gilleade and Alan Dix. 2004. Using frustration in the design of adaptive videogames. *Proceedings of the 2004 ACM SIGCHI International Conference on Advances in computer entertainment technology, ACM*, 228–232. <http://doi.org/10.1145/1067343.1067372>
  28. Hamilton A. Hernandez, T. C. Graham, Darcy Fehlings, et al. 2012. Design of an exergaming station for children with cerebral palsy. *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*, 2619–2628. Retrieved May 24, 2013 from <http://dl.acm.org/citation.cfm?id=2208652>
  29. Johan Huizinga. 2008. *Homo ludens: proeve einer bepaling van het spel-element der cultuur*. Amsterdam University Press.
  30. Robin Hunicke. 2005. The case for dynamic difficulty adjustment in games. *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology*, 429–433.
  31. Marshall G. Jones. 1998. Creating Electronic Learning Environments: Games, Flow, and the User Interface. Retrieved September 17, 2015 from <http://eric.ed.gov/?id=ED423842>
  32. Arvid Kappas. 2010. Smile when you read this, whether you like it or not: Conceptual challenges to affect detection. *IEEE Transactions on Affective Computing*, 1, 1: 38–41.
  33. Pamela M. Kato. 2012. Evaluating Efficacy and Validating Games for Health. *Games for Health Journal* 1, 1: 74–76. <http://doi.org/10.1089/g4h.2012.1017>
  34. K. Kuikkaniemi, T. Laitinen, M. Turpeinen, T. Saari, I. Kosunen, and N. Ravaja. 2010. The influence of implicit and explicit biofeedback in first-person shooter games. *Proceedings of the 28th international conference on Human factors in computing systems*, 859–868. Retrieved August 17, 2012 from <http://dl.acm.org/citation.cfm?id=1753453>
  35. E McAuley, T Duncan, and V V Tammen. 1989. Psychometric properties of the Intrinsic Motivation Inventory in a competitive sport setting: a confirmatory factor analysis. *Research quarterly for exercise and sport* 60, 1: 48–58.
  36. Giovanni B. Moneta. 2012. On the Measurement and Conceptualization of Flow. In *Advances in Flow Research*, Stefan Engeser (ed.). Springer New York, 23–50. Retrieved September 17, 2015 from [http://link.springer.com/chapter/10.1007/978-1-4614-2359-1\\_2](http://link.springer.com/chapter/10.1007/978-1-4614-2359-1_2)
  37. A. Quilici. 1994. Forming user models by understanding user feedback. *User Modeling and User-Adapted Interaction* 3, 4: 321–358.
  38. Leonard Reinecke, Ron Tamborini, Matthew Grizzard, Robert Lewis, Allison Eden, and Nicholas David Bowman. 2012. Characterizing Mood Management as Need Satisfaction: The Effects of Intrinsic Needs on Selective Exposure and Mood Repair. *Journal of Communication* 62, 3: 437–453. <http://doi.org/10.1111/j.1460-2466.2012.01649.x>
  39. Scott Rigby and Richard Ryan. 2007. *The Player Experience of Need Satisfaction(PENS): an applied model and methodology for understanding key components of the player experience*.
  40. Scott Rigby and Richard Ryan. 2011. *Glued to Games: How Video Games Draw Us In and Hold Us Spellbound*. Praeger.
  41. Ute Ritterfeld, Michael Cody, and Peter Vorderer. 2009. *Serious Games: Mechanisms and Effects*. Routledge.
  42. Richard M. Ryan, C. Scott Rigby, and Andrew Przybylski. 2006. The Motivational Pull of Video Games: A Self-Determination Theory Approach. *Motivation and Emotion* 30, 4: 344–360. <http://doi.org/10.1007/s11031-006-9051-8>
  43. Katie Salen and Eric Zimmerman. 2003. *Rules of play: game design fundamentals*. MIT Press, Cambridge, Mass.
  44. Rosa García Sánchez, Alasdair G. Thin, Jannicke Baalsrud Hauge, et al. 2012. Value Propositions for Serious Games in Health and Well-Being. In *Serious*

- Games Development and Applications*, Minhua Ma, Manuel Fradinho Oliveira, Jannicke Baalsrud Hauge, Heiko Duin and Klaus-Dieter Thoben (eds.). Springer Berlin Heidelberg, 150–157. Retrieved July 5, 2013 from [http://link.springer.com/chapter/10.1007/978-3-642-33687-4\\_12](http://link.springer.com/chapter/10.1007/978-3-642-33687-4_12)
45. Thomas B. Sheridan. 2001. Rumination on automation, 1998. *Annual Reviews in Control* 25: 89–97. [http://doi.org/10.1016/S1367-5788\(01\)00009-8](http://doi.org/10.1016/S1367-5788(01)00009-8)
46. Jan David Smeddinck, Marc Herrlich, and Rainer Malaka. 2015. Exergames for Physiotherapy and Rehabilitation: A Medium-term Situated Study of Motivational Aspects and Impact on Functional Reach. *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, ACM, 4143–4146. <http://doi.org/10.1145/2702123.2702598>
47. Jan Smeddinck, Sandra Siegel, and Marc Herrlich. 2013. Adaptive Difficulty in Exergames for Parkinson's disease Patients. *Proceedings of Graphics Interface 2013*.
48. Penelope Sweetser and Peta Wyeth. 2005. GameFlow: A Model for Evaluating Player Enjoyment in Games. *Comput. Entertain.* 3, 3: 3–3. <http://doi.org/10.1145/1077246.1077253>
49. Jane Webster and And Others. 1993. The Dimensionality and Correlates of Flow in Human-Computer Interactions. *Computers in Human Behavior* 9, 4: 411–26.
50. Philip Zigoris and Yi Zhang. 2006. Bayesian adaptive user profiling with explicit & implicit feedback. *Proceedings of the 15th ACM international conference on Information and knowledge management*, ACM, 397–404. <http://doi.org/10.1145/1183614.1183672>

# Game and Player Feature Selection for Entertainment Capture

Georgios N. Yannakakis\* and John Hallam†

Mærsk Mc-Kinney Møller Institute

University of Southern Denmark

Campusvej 55, DK-5230, Odense

{georgios\*;john†}@mimi.sdu.dk

**Abstract**— The notion of constructing a metric of the degree to which a player enjoys a given game has been presented previously. In this paper, we attempt to construct such metric models of children's 'fun' when playing the Bug Smasher game on the Playware platform. First, a set of numerical features derived from a child's interaction with the Playware hardware is presented. Then the Sequential Forward Selection and the n-Best feature selection algorithms are employed together with a function approximator based on an artificial neural network to construct feature sets and function that model the child's notion of 'fun' for this game. Performance of the model is evaluated by the degree to which the preferences predicted by the model match those expressed by the children in a survey experiment.

The results show that an effective model can be constructed using these techniques and that the Sequential Forward Selection method performs better in this task than n-Best. The model reveals differing preferences for game parameters between children who react fast to game events and those who react slowly. The limitations and the use of the methodology as an effective adaptive mechanism to entertainment augmentation are discussed.

**Keywords:** Entertainment modeling, intelligent interactive playgrounds, neuro-evolution.

## I. INTRODUCTION

Cognitive modeling projects significant potential within digital interactive entertainment systems (such as computer games). Being able to model the level of user (gamer) engagement or satisfaction in real-time can provide insights to the appropriate AI methodology for enhancing the quality of playing experience [1] and furthermore be used to adjust digital entertainment environments according to individual user preferences.

The 'Playware' [2] intelligent interactive physical playground attempts to combine the advantages of both computer games and traditional playgrounds. On one hand, computer games keep children (among others) engaged more than other digital media because of their high degree of interactivity and the freedom for the child to develop and play a role within a fantasy world which is created during play [3]. On the other hand, traditional playgrounds offer the advantage of physical play, which furthermore improves the child's health condition, augment children's ability to engage in social and fantasy play [4], [5] and provide the freedom for children to generate their own rules for their own developed games. Experiments with children playing Playware games will be presented in this paper.

Following from the reported successful entertainment capture in physical interactive games [6], a further endeavor

on capturing player satisfaction during gameplay (i.e. entertainment modeling) through more extensive methodology and experiments with children players is presented in this paper. As in [6], this is achieved by following the theoretical principles of Malone's intrinsic qualitative factors for engaging gameplay [3], namely *challenge*, *curiosity* and *fantasy*. Quantitative measures for challenge and curiosity are used from previous studies on quantitative reported entertainment capture [6] in the Playware playground.

A mapping between the aforementioned factors (game features) and the children's notion of 'fun' or entertainment (the two terms are used interchangeably herein) is derived using a game developed on the Playware playground as a test-bed. Each player's individual characteristics (player features), such as response time and foot pressure, are recorded during each game. Feedforward artificial neural networks (ANNs) are trained using artificial evolution on this gameplay experimental data to construct a function mapping the examined game features and player features to the reported player satisfaction preferences. The n-Best (nBest) and the Sequential Forward Selection (SFS) [7] feature selection methods are used to extract the minimal subset of game and player features to be included in the ANN model.

Single feature experiments demonstrate that the average response time of children interacting with the playground is the feature that yields the best (highest-performing) mapping between game and player features and children's expressed preferences on entertainment. This result is consistent with the reported impact (i.e. significant linear correlation) of the average response time on reported entertainment in Playware games [6]. When more than one feature is examined, the SFS method generates feature subsets performing better overall than the subsets generated by the nBest method. More specifically, it finds a set of four features that yields the highest performance in matching opponent's and player's behavior to children's perceived entertainment. Player features include the player's average response time with the playground, the variance of the pressure force instances on the playground and the number of interactions with the playground. The game feature included in the most accurate model of player satisfaction obtained is the level of curiosity generated by the game opponents. Analysis of the obtained model shows that different children (classified by their response time) have different requirements on the levels of the curiosity factor for the game to be judged entertaining.

The work reported here is novel in that it isolates game and player features attributed to reported entertainment in physically demanding games and demonstrates a way of constructing a subjective model (a predictor of user preferences) of reported entertainment grounded in statistical features obtained from child-game interaction. The limitations of the proposed methodology and its extensibility to other genres of digital entertainment are discussed. Its generic use as an efficient baseline for capturing reported entertainment in physical interactive games in real-time is also outlined.

## II. ENTERTAINMENT CAPTURE

There have been several psychological studies to identify what is ‘fun’ in a game and what engages people playing computer games. Theoretical approaches include Malone’s principles of intrinsic qualitative factors for engaging game play [3], namely challenge, curiosity and fantasy as well as the well-known concepts of the theory of flow [8] incorporated in computer games as a model for evaluating player enjoyment, namely *GameFlow* [9]. A comprehensive review of the literature on qualitative approaches for modeling player enjoyment demonstrates a tendency of overlapping with Malone’s and Csikszentmihalyi’s foundational concepts. Many of these approaches are based on Lazzaro’s ‘fun’ clustering which uses four entertainment factors based on facial expressions and data obtained from game surveys on players [10]: hard fun, easy fun, altered states and socialization. Koster’s [11] theory of fun, which is primarily inspired by Lazzaro’s four factors, defines ‘fun’ as the act of mastering the game mentally. An alternative approach to fun capture is presented in [12] where fun is composed of three dimensions: durability, engagement and expectations.

Vorderer et al. [13] present a quantitative analysis of the impact of competition (i.e. challenge) on entertainment and identify challenge as the most important determinant of the enjoyment perceived by video game (*Tomb Raider*) players. They claim that a successful completion of a task generates sympathetic arousal, especially when the challenge of the task matches the player’s abilities. According to Choi et al. [14], challenge and satisfaction appear as independent processes, in contrast to the views of Malone [3] and Yannakakis et al. [6] where satisfaction derives from the appropriate level of challenge and other game components.

Iida’s work on metrics of entertainment in board games was the first attempt in the area of quantitative ‘fun’ modeling. He introduced a general metric of entertainment for variants of chess games depending on average game length and possible moves [15]. Other work in the field of quantitative entertainment capture is based on the hypothesis that the player-opponent interaction — rather than the audiovisual features, the context or the genre of the game — is the property that contributes the majority of the quality features of entertainment in a computer game [16]. Based on this fundamental assumption, a metric for measuring the real time entertainment value of predator/prey games was designed, and established as efficient and reliable by validation against human judgement [17], [18]. Further studies by Yannakakis

and Hallam [19] have shown that Artificial Neural Networks (ANN) and fuzzy neural networks can extract a better estimator of player satisfaction than a human-designed one, given appropriate estimators of the challenge and curiosity of the game and data on human players’ preferences.

A step further to entertainment capture is towards games of richer human-computer interaction and affect recognizers which are able to identify correlations between physiological signals and the human notion of entertainment. Experiments by Yannakakis et al. [20] have already shown a significant effect of children’s average heart rate on children’s reported entertainment in action games played in interactive physical playgrounds. Moreover, Rani et al. [21] propose a methodology for detecting anxiety level of the player and appropriately adjusting the level of challenge (e.g. speed) in the game of ‘Pong’. Physiological state (hear-rate, galvanic skin response) prediction models have also been proposed for potential entertainment augmentation in computer games [22]. Similar work in adjusting a game’s difficulty include endeavors through reinforcement learning [23], genetic algorithms [24], probabilistic models [25] and dynamic scripting [26]. However, the aforementioned attempts are based on the assumption that challenge is the only factor that contributes to enjoyable gaming experiences while results reported have not been cross-verified by human players.

Following the theoretical principles reported from Malone [3], Koster [11] and Yannakakis [18], this paper is primarily focused on the contributions of game opponents’ behavior to the real-time entertainment value of the game. We argue that among the three dimensions of ‘fun’ (durability, engagement, expectations) defined in [12] it is only *engagement* that is affected by the opponent since both *durability* and *expectations* are based primarily on the game design *per se*. Given a successful interactive game design that yields high expectations and durability, we only focus on the level of engagement that generates ‘fun’ (entertainment). However, instead of being based on empirical observations of children’s entertainment, the work presented here uses quantitative measures for Malone’s entertainment factors of challenge and curiosity (as introduced in [6]). On that basis, a mapping between the two aforementioned factors, children’s play recorded features and their expressed preferences is constructed using experimental data obtained from a survey experiment with children playing with Playware playground (see Section III).

## III. PLAYWARE PLAYGROUND

The Playware [2] prototype playground consists of several building blocks (i.e. tangible tiles) that allow for the game designer (e.g. the child) to develop a significant number of different games within the same platform. The overall technological concept of Playware is based on embodied AI [27] where intelligent physical identities (tiles) incorporate processing power, communication, input and output, focusing on the role of the morphology-intelligence interplay in developing game platforms. See [6], [2] for further details on Playware playground.



Fig. 1. A child playing the Bug-Smasher game.

#### A. Bug-Smasher Game

The test-bed game used for the experiments presented here is called ‘Bug-Smasher’. The game is developed on a  $6 \times 6$  square tile topology (see Fig. 1). During the game, different ‘bugs’ (colored lights) appear on the game surface and disappear sequentially after a short period of time by turning a tile’s light on and off respectively. A bug’s position is picked randomly according to the predefined level of the bugs’ spatial diversity. Spatial diversity is measured by the entropy ( $H$ ) of the bug-visited tiles.

The child’s goal is to smash as many bugs as possible by stepping on the lighted tiles. Bug-smasher has been used as a test-bed in previous work; further details can be found in [6], [20] and [28].

#### IV. EXPERIMENTAL DATA

The Bug-Smasher game has been used to acquire data of children’s judgement on entertainment. Three states (‘Low’, ‘Average’, and ‘High’) are used for each of the two entertainment factors of challenge and curiosity summing up to 9 different game states. The fantasy factor is not investigated through this survey since the focus of this paper is on the opponent (bug) contribution to entertainment. See [28] for fantasy’s positive impact on entertainment in Bug-Smasher.

We consider (as in [6]) the speed ( $S$  — in  $\text{sec}^{-1}$ ) that the bugs appear and disappear from the game and their spatial diversity ( $H$ ) on the game’s plane as appropriate measures to represent the level of challenge and the level of curiosity (unpredictability) respectively [3] during gameplay. The former provides a notion for a goal whose attainment is uncertain and the latter effectively portrays a notion of unpredictability in the subsequent events of the game — the higher the  $H$  value the higher the bug appearance unpredictability and therefore the higher the curiosity.

Seventy two normal-weighted (based on their body mass index) children whose age covered a range between 8 and 10 years participated in an experiment. By experimental design, each subject plays against two of the selected game

states in all permutations of pairs. The number of children participated in the experiment is derived from  $2 \cdot C_2^9 = 72$  being twice the required number of all combinations of 2 out of 9 game states. In this experiment, each subject plays two games ( $A$  and  $B$ ) for 90 seconds each; the two games differ in the levels of one or both entertainment factors of challenge and curiosity. Each time a pair of games is finished, the child is asked whether the first game was more ‘fun’ (see [12] for terminology used in experiments with children) than the second game i.e. whether  $A$  or  $B$  generated a more entertaining game. The 2-alternative forced choice (2-AFC) approach is used since it offers several advantages for a subjective entertainment capture: it minimizes the assumptions made about children’s notions of “fun” and allows a fair comparison between the answers of different children. Since our focus is to construct a model relating reported entertainment preferences to game and player features that generalises over the reports of different children 2-AFC is preferred to a ranking approach [29]. Note also that, children are not interviewed but are asked to fill in a questionnaire, minimizing the interviewing effects reported in [29].

The child’s answers are used to guide the training of an ANN model of reported entertainment (see Section V). In order to minimize any potential order effects we let each subject play the aforementioned games in the inverse order too. Statistical analysis of the subjects’ answers shows that no significant order effect occurs ( $r_c = -0.102$ , p-value= 0.224). The reported insignificant order effect also, in part, demonstrate that effects such as a child’s possible preference for the very first game played and the interplay between reported entertainment and familiarity with the game are statistically insignificant. The total number of game pairs played equals 144; however, data from 137 game pairs are used due to hardware (communication ports) failure during seven games.

Since, with the current implementation of the Playware platform, the only input to the system is through a Force Sensing Resistor (FSR) sensor, quantitative individual playing characteristics can only be based on three measurable features: the state (position and LEDs color) of a pressed tile, the time that a tile-press event took place and the pressure force on a pressed tile. Pressed tile events are recorded in real-time and a selection of nine personalized (individual) player features are calculated for each child. These include the number of smashed bugs over the total number of bugs appeared  $P$  (i.e. child’s score); the number of interactions with the game environment  $N_I$ ; the average and the variance of the response times ( $E\{r_t\}, \sigma^2\{r_t\}$ ); the average and the variance of the distance between the pressed tile and the bugs appearing on the game ( $E\{D_b\}, \sigma^2\{D_b\}$ ); the average and the variance of the pressure recorded from the FSR sensor  $E\{p\}, \sigma^2\{p\}$ ); and the entropy of the tiles that the child visited  $H_C$ .

#### A. Statistical Analysis

The aim of the statistical analysis presented here is to identify statistically significant correlations between children’s

notion of entertainment and any of the aforementioned individual player features and/or the quantitative entertainment factors (game features): challenge and curiosity. For this purpose the following null hypothesis is formed: The correlation between observed children judgement of entertainment and recorded player and game features, as far as the different game states are concerned, is a result of randomness. The test statistic is obtained through  $c(\vec{z}) = \sum_{i=1}^{N_s} \{z_i/N_s\}$ , where  $N_s$  is the total number of game pairs played ( $N_s = 137$ ) and  $z_i = 1$ , if the subject chooses as the more entertaining game the one with the larger value of the examined feature and  $z_i = -1$ , if the subject chooses the other game in the game pair  $i$ .

The obtained significant — significance equals 5%, high significance equals 1% in this paper — effects of the selected features on reported entertainment are:  $N_I$  ( $c(\vec{z}) = 0.1678$ , p-value = 0.0298),  $E\{r_t\}$  ( $c(\vec{z}) = -0.2262$ , p-value = 0.0050),  $\sigma^2\{r_t\}$  ( $c(\vec{z}) = -0.1532$ , p-value = 0.0435),  $E\{p\}$  ( $c(\vec{z}) = 0.1678$ , p-value = 0.0298) and  $\sigma^2\{p\}$  ( $c(\vec{z}) = 0.1970$  p-value = 0.0129). These effects appear to be commonsensical since the Bug-Smasher game belongs to the genre of action physical games where the level of engagement of the user tends to have a significant effect on the number of interactions and the reaction time of the player [30]. In Bug-Smasher, the more a child is entertained the more ( $N_I$ ) and harder ( $E\{p\}$ ) she/he tends to interact with the game platform. This behavior generates lower average response time ( $E\{r_t\}$ ) and higher average pressure on the tiles ( $E\{p\}$ ). Moreover, it appears that the variability of the aforementioned individual characteristics ( $\sigma^2\{r_t\}$ ,  $\sigma^2\{p\}$ ) does have an effect on reported entertainment too. The obtained highly significant effect of  $E\{r_t\}$  is consistent with previous experiments on the Bug-Smasher game [6].

On the other hand it appears that reported entertainment cannot be objectively modeled according to the levels of challenge ( $c(\vec{z}) = 0.0$ , p-value = 0.5382) and curiosity ( $c(\vec{z}) = 0.0909$ , p-value = 0.1448) since there exists a level of personalization which has to be included as a factor in entertainment modeling. The feature selection procedure presented in Section VI allows the designer to choose specific individual player features that can successfully map between children's behavior, game features and reported entertainment.

## V. EVOLVING ANN

The proposed approach to entertainment modeling is based on selecting a minimal subset (see Section VI) of game and player features and constructing a quantitative user model that predicts the children's reported entertainment preferences. For this purpose, a fully-connected feedforward ANN for learning the relation between the selected game and player features (ANN inputs) and the "entertainment value" (ANN output) of a game is presented. The assumption is that the entertainment value  $y$  of a given game is an unknown function of player and game features which the ANN will learn. The children's expressed preferences constrain but do not specify the values of  $y$  for individual games but we

assume that the child's expressed preferences are consistent. Since there are no prescribed target outputs for the learning problem (i.e. no differentiable output error function), ANN training algorithms such as back-propagation are inapplicable. Learning is achieved through artificial evolution [31] and is described in Section V-A.

The sigmoid function is employed at each neuron, the connection weights take values from -5 to 5 to match with input values normalized into [0, 1] before they are entered into the ANN. In an attempt to minimize the controller's size, it was determined that a single hidden-layered ANN architecture, containing 20 hidden neurons, is capable of successfully obtaining solutions of high fitness. This was determined by considering the performance of ANN architectures with up to two hidden layers containing up to 30 hidden neurons each.

### A. Genetic Algorithm

A generational genetic algorithm (GA) [32] is implemented, which uses a fitness function that measures the difference between the children's reported preferences of entertainment and the model output value  $y$ . The ANN is itself evolved. In the algorithm presented here, the ANN topology is fixed and the GA chromosome is a vector of ANN connection weights. The algorithm is described briefly in this section since it has previously presented in [6].

A population of  $N$  ( $N$  is 1000 in this paper) networks is initialized randomly. Initial real values that lie within [-5, 5] for their connection weights are picked randomly from a uniform distribution. Then, at each generation: (a) Each member (neural network) of the population is given two  $n_i$ -tuple (where  $n_i$  is the number of game or player features) values one for opponent/game  $A$  and one for opponent/game  $B$  for each pair  $j$  of games played in the survey experiment ( $N_s = 137$ ) — see [6] for further details. In each case it returns two output values, representing the level of 'fun' in each game, namely  $y_{j,A}$  and  $y_{j,B}$ . (b) Each member  $i$  of the population is evaluated via a fitness function  $f_i$  that promotes the matching between ANN outputs ( $y$ ) and children's reported answers (see [6]). A high fitness results if the ranking of  $y_{j,A}$  and  $y_{j,B}$  matches the expressed preference of the children for each game pair  $j$ . (c) A fitness-proportional selection method is used. (d) Montana and Davis [33] crossover and Gaussian mutation are applied (see [6]).

The algorithm is terminated when either a good solution is found ( $f > 0.95f_{max}$ ; where  $f_{max}$  is the maximum fitness) or a large number of generations  $g$  is completed ( $g = 10000$ ).

## VI. FEATURE SELECTION

There are two different feature selection schemes applied and compared in this paper. Given both the individual player features and the game features presented in section IV the *n Best Features Selection* (nBest) and the *Sequential Forward Selection* (SFS) methods are applied. The nBest selection method picks the  $n$  individually best features (with regards to a performance function) from the feature subset. The SFS method, by contrast, is a bottom-up search procedure where one feature is added at a time to the current feature set.

The feature to be added is selected from the subset of the remaining features so that the new feature set generates the maximum value of the performance function over all candidate features for addition [7].

The SFS method is used since it has been successfully applied in a wide variety of feature selection problems yielding high performance values with minimal feature subsets; see [34], for example, for further discussion and application to the classification problem of process identification in resistance spot welding. On the other hand, the nBest method is used for comparative purposes being the most popular technique for feature selection. Features selected by each method constitute the input vector of the evolving ANN. The feature selection procedure followed here evaluates the usability of each one of the features available and obtains the minimal feature subset that performs best in the classification between games reported as entertaining and games reported as non-entertaining (see Section V).

To evaluate the performance of each feature subset the available data is randomly divided into training and validation data sets consisting of 2/3 and 1/3 of the data respectively. The performance of an ANN model is measured through the average classification accuracy of the ANN in three independent runs using the leave-one-out cross-validation technique on the training and validation data sets. Since we are interested in the minimal feature subset that yields the highest performance we terminate the feature selection procedure (nBest or SFS) when an added feature yields equal or lower validation performance than the performance obtained without it.

#### A. Single Feature Performance

The experiment presented here tests the validation performance of single individual player and game features. Given the selected feature (ANN input), ANNs are evolved by following the approach presented in Section V-A and evaluated through the leave-one-out cross-validation method (see Section VI). The training and validation performance of each of the individual player and game features are presented in Table I where features are ranked by validation performance.

The impact of the recorded response times ( $r_t$ ) is demonstrated in Table I; both the average and the variance of these values generate the highest cross-validation performances (see also [6] for the impact of  $E\{r_t\}$  on reported entertainment in the Bug-Smasher game). Results obtained show the incapability of a single feature to successfully model reported entertainment in Bug-Smasher. Given that the best performed feature ( $E\{r_t\}$ ) yields a cross-validation performance of 62.22% it becomes apparent that more features are required to effectively model children's notion of entertainment. Moreover, it appears that results presented in Table I are consistent with the correlates of reported entertainment presented in Section IV-A. In fact, three out of four best features of Table I yield statistically significant effects on reported entertainment (see Section IV-A).

TABLE I

TRAINING AND VALIDATION PERFORMANCE OF INDIVIDUAL PLAYER AND GAME FEATURES.  $E\{r_t\}$  AND  $\sigma^2\{r_t\}$  IS THE AVERAGE AND THE VARIANCE OF THE RESPONSE TIME RESPECTIVELY;  $\sigma^2\{D_b\}$  IS THE VARIANCE OF THE DISTANCES BETWEEN THE PRESSED TILE AND THE BUGS APPEARING ON THE GAME;  $N_I$  IS THE TOTAL NUMBER OF INTERACTIONS;  $H$  IS THE QUANTITATIVE MEANS FOR THE GAME CONTROLLABLE FEATURE OF CURIOSITY;  $E\{p\}$  IS THE AVERAGE PRESSURE FORCE RECORDED FROM THE FSR SENSOR;  $H_C$  IS THE ENTROPY OF THE TILES THAT THE CHILD VISITED;  $\sigma^2\{p\}$  IS THE VARIANCE OF THE PRESSURE FORCES RECORDED FROM THE FSR SENSOR;  $E\{D_b\}$  IS THE AVERAGE DISTANCE BETWEEN THE PRESSED TILE AND THE BUGS APPEARING ON THE GAME;  $S$  IS THE QUANTITATIVE MEANS FOR THE GAME CONTROLLABLE FEATURE OF CHALLENGE AND  $P$  IS THE TOTAL NUMBER OF SMASHED BUGS.

| Feature           | Training Performance (%) | Validation Performance (%) |
|-------------------|--------------------------|----------------------------|
| $E\{r_t\}$        | 69.47                    | 62.22                      |
| $\sigma^2\{r_t\}$ | 67.91                    | 61.11                      |
| $\sigma^2\{D_b\}$ | 68.54                    | 56.67                      |
| $N_I$             | 66.36                    | 56.67                      |
| $H$               | 66.04                    | 55.56                      |
| $E\{p\}$          | 64.17                    | 53.33                      |
| $H_C$             | 59.81                    | 53.33                      |
| $\sigma^2\{p\}$   | 66.73                    | 51.11                      |
| $E\{D_b\}$        | 65.42.                   | 51.11                      |
| $S$               | 43.93                    | 46.67                      |
| $P$               | 65.42                    | 43.33                      |

#### B. More Features: Selection Method Comparison

This section presents experiments for finding the minimal feature subset that yields the highest classification performance in matching the ANNs output with children's reported answers on entertainment in unknown data (validation data set). For this purpose, the two feature selection methods described in Section VI are applied and compared. The initial subset (ANN input) for both methods includes the feature that performs best in the single feature experiment:  $E\{r_t\}$ . ANNs are evolved by following the approach presented in Section V-A. The data is partitioned in training (2/3 of total data) and validation (1/3 of total data) portions and the leave-one-out cross-validation technique is used to obtain the classification performance of the ANNs.

Table II presents the above-mentioned comparative study between nBest and SFS. SFS appears to generate feature subsets that yield higher validation performance than feature subsets generated by nBest. The best cross-validation performance (77.77%; average of 70%, 73.33% and 90%) is achieved when the ANN input contains  $E\{r_t\}$ ,  $\sigma^2\{p\}$ ,  $H$  and  $N_I$ . The binomial-distributed probability of this performance to occur at random is 0.0019 demonstrating statistical significance and providing evidence for this solution's robustness. Note that, challenge is absent from the obtained feature subset indicating that the spatial diversity of the bugs (curiosity) has a higher impact on children's reported

entertainment than the speed of the game (challenge).

Difficulties in obtaining higher classification accuracy are found in experimental noise in both the recorded features and the children's answers on self reports. Even though comparative fun analysis is a reliable and established method for capturing reported entertainment in computer [18] and mixed-reality [6] games, it generates a significant amount of uncertainty in subjects' reported answers. Uncertainty appears when the two games played are not significantly different with regards to the entertainment value they generate for the player and therefore cannot be distinguished.

TABLE II

VALIDATION PERFORMANCE OF INDIVIDUAL PLAYER AND GAME FEATURES AND THEIR RESPECTIVE BINOMIAL DISTRIBUTED P-VALUES.

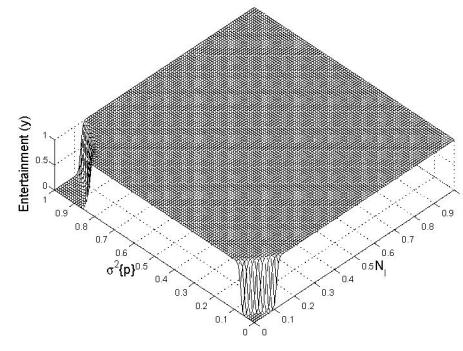
$E\{r_t\}$  AND  $\sigma^2\{r_t\}$  IS THE AVERAGE AND THE VARIANCE OF THE RESPONSE TIME RESPECTIVELY;  $\sigma^2\{p\}$  IS THE VARIANCE OF THE PRESSURE FORCES RECORDED FROM THE FSR SENSOR;  $\sigma^2\{D_b\}$  IS THE VARIANCE OF THE DISTANCES BETWEEN THE PRESSED TILE AND THE BUGS APPEARING ON THE GAME;  $H$  IS THE QUANTITATIVE METRIC OF CURIOSITY AND  $N_I$  IS THE TOTAL NUMBER OF INTERACTIONS.

| Features          | nBest | $p_{nBest}$ | Features          | SFS          | $p_{SF}$      |
|-------------------|-------|-------------|-------------------|--------------|---------------|
| $E\{r_t\}$        | 62.22 | 0.1270      | $E\{r_t\}$        | 62.22        | 0.1270        |
| $\sigma^2\{r_t\}$ | 58.88 | 0.2179      | $\sigma^2\{p\}$   | 67.77        | 0.0400        |
| $\sigma^2\{D_b\}$ | 44.44 | 0.2551      | $H$               | 68.88        | 0.0307        |
| $N_I$             | 46.67 | 0.4277      | $N_I$             | <b>77.77</b> | <b>0.0019</b> |
| $H$               | 52.22 | 0.4759      | $\sigma^2\{r_t\}$ | 63.33        | 0.1002        |

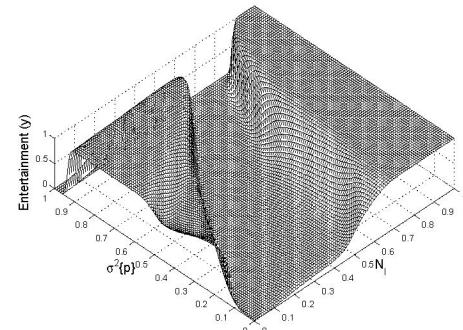
For reasons of space, only the feature subset  $\{E\{r_t\}, \sigma^2\{p\}, N_I, H\}$  with the highest validation performance (90.00%, in one of the three learning attempts) is presented in this paper. Note that, the qualitative features of the surfaces plotted in Fig. 2 appeared in all three different learning attempts of the cross-validation procedure for this feature subset.

Fig. 2 illustrates the trained ANN output with regards to  $\sigma^2\{p\}$  and  $N_I$  for six points in the  $(E\{r_t\}, H)$  search space. These values constitute the combinations of two  $E\{r_t\}$  states (0 and 1 named *Fast* and *Slow* respectively), and the three states used for  $H$  (0.33, 0.66 and 1 named *Low*, *Average* and *High* respectively). The above presentation helps towards interpreting the mapping between  $\sigma^2\{p\}$ ,  $N_I$  and reported entertainment according to how fast children react with the playground and the level of curiosity.

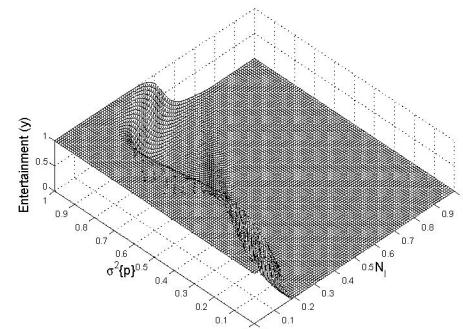
As seen from Fig. 2, fast children ( $E\{r_t\} = 0$ ) appear to enjoy average and high curiosity values except when high  $N_I$  values are combined with low values of  $\sigma^2\{p\}$  (see Fig. 2(e) and Fig. 2(f)). Fast children's preference for low levels of curiosity is met only when their behavior combines low values of  $N_I$  and high values of  $\sigma^2\{p\}$  (see Fig. 2(d)). On the other hand, slow children appear to prefer low curiosity levels except when the  $N_I$  value they generate is low and combined with either very high or very low  $\sigma^2\{p\}$  values (see Fig. 2(a)). Average curiosity levels are preferred by slow children in many fewer cases; that is when their  $N_I$  value



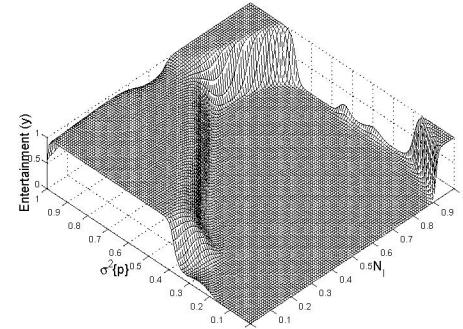
(a)  $E\{r_t\} = 1.0$  (Slow),  $H = 0.33$  (Low)



(b)  $E\{r_t\} = 1.0$  (Slow),  $H = 0.66$  (Average)



(c)  $E\{r_t\} = 1.0$  (Slow),  $H = 1.0$  (High)



(d)  $E\{r_t\} = 0.0$  (Fast),  $H = 0.33$  (Low)

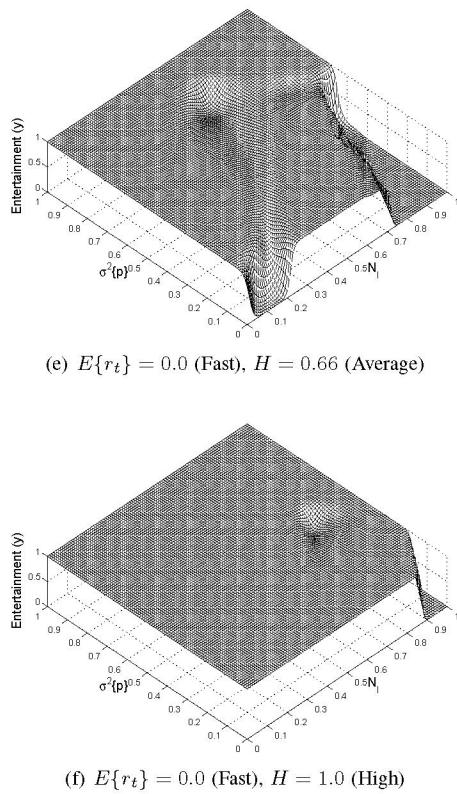


Fig. 2. The trained ANN ( $f = 95.85$ ) that yields the highest validation performance (90.00%): ANN output  $y$  (entertainment value) with regards to  $\sigma^2\{p\}$  and  $N_I$  for all six combinations of two  $E\{r_t\}$  states (Slow, Fast) and three  $H$  states (Low, Average, High).

is low and  $\sigma^2\{p\}$  value is high or when their  $N_I$  value is high and  $\sigma^2\{p\}$  value is low (see Fig. 2(b)). Finally, high curiosity is rarely preferred by slow children and this occurs only when their  $N_I$  values are low independently of their  $\sigma^2\{p\}$  value (Fig. 2(c)).

The obtained effects of curiosity in reported entertainment are consistent, in part, with previous studies on the Bug-Smasher game [6]. In that study the relation between challenge, curiosity and average response time was reported through a lower scale experiment of 28 children. It was found that fast children liked games independently of curiosity whereas children reacting slowly with the playground preferred games of low curiosity levels.

## VII. CONCLUSIONS & DISCUSSION

This paper introduced feature selection methods for obtaining minimal feature subsets that successfully model children's notion of entertainment through the Bug-Smasher game played on the Playware playground. More specifically, the nBest and the SFS feature selection methods were applied and compared demonstrating the ability of SFS in finding feature subsets that yield higher validation performance.

The fittest ANN solution presented derives from a feature subset of four features:  $\{E\{r_t\}, \sigma^2\{p\}, N_I, H\}$ . Experi-

ments with additional features (inputs of the ANN) could not improve the model's validation performance. This model manages to map between children's average response time, the variance of their force pressure on the tiles, the number of interactions with the playground, the game feature of curiosity and the children's notion of gameplay entertainment with a cross-validation accuracy of 77.77% (binomial-distributed p-value = 0.0019). The main reason for not obtaining a higher cross-validation performance appears to be the experimental noise existent in the self-reports designed for comparative entertainment (fun) analysis. Moreover, the learned mapping between  $\{E\{r_t\}, \sigma^2\{p\}, N_I, H\}$  and children's notion of entertainment showed that, in general, fast responding children show a preference for high curiosity games whereas slow responding children tend to prefer games of low curiosity. The obtained results are consistent with previous work on the impact of the factors of challenge and curiosity and the average response time in Playware games [6].

The main limitation of the proposed approach lies within the complexity of entertainment as a mental state. The generated  $y$  value cannot be regarded as a mental affective state approximator but as a correlate of expressed children's preferences on entertainment. However, this correlate serves the purposes of this work well as far as entertainment modeling is concerned. In addition, Malone's entertainment factor of fantasy is omitted from the results in this paper since the focus is on the contribution of the opponent behaviors to the generation of entertainment; however, fantasy's positive impact on reported entertainment has been reported in a previous study [28].

Even though the comparative fun protocol (2-AFC) used serves well the purpose of this work, a 4-alternative forced choice (4-AFC) approach is considered for future experiment protocol design. Children will be able to choose among the following alternatives: one game is more "fun" than the other (2-AFC), both games are equally "fun", neither game was "fun". This protocol would provide more information for the machine learning process and eliminate the noise generated by 2-AFC.

The entertainment modeling approach presented here demonstrates generality over the majority of action games created with Playware since the quantitative measures of challenge and curiosity are estimated through the generic features of speed and spatial diversity of the opponent on the game's surface. Thus, these or similar measures could be used to adjust player satisfaction in any future game development on the Playware tiles. However, each game demonstrates individual entertainment features that might need to be extracted and added on the proposed measures and therefore, more games of the same and/or other genres need to be tested to cross-validate this hypothesis.

The proposed approach can be used for adaptation of the game opponents (e.g. bugs) according to the player's individual playing style, based on reaction time, recorded pressure on tiles and amount of interactions, and as far as the curiosity factor of entertainment is concerned. Given the

real-time average response time of a child, the variance of his/her pressure forces on the tiles and the number of times he/she interacts with the environment, the partial derivative of the model output  $\partial y / \partial H$  can be used to appropriately adjust the level of entropy (curiosity) of the opponent for the entertainment value  $y$  to be augmented. Such a direction constitutes an example of future work on Playware and computer games. The level of engagement or motivation of the user/player/gamer of such interactive environments may be identified and augmented by the use of the presented approaches.

#### ACKNOWLEDGMENTS

The authors would like to thank Henrik Jørgensen and all children of Henriette Hørlücks and Rosengårdskolen Schools, Odense, Denmark that participated in the experiments.

The tiles were designed by C. Isaksen from Isaksen Design and parts of their hardware and software implementation were collectively done by A. Derakhshan, F. Hammer, T. Klitbo and J. Nielsen. KOMPAN, Mads Clausen Institute, and Danfoss Universe also participated in the development of the tiles.

This work was in part supported by the Danish National Research Council (project no: 274-05-0511).

#### REFERENCES

- [1] G. N. Yannakakis and J. Hallam, "A scheme for creating digital entertainment with substance," in *Proceedings of the Workshop on Reasoning, Representation, and Learning in Computer Games, 19th International Joint Conference on Artificial Intelligence (IJCAI)*, August 2005, pp. 119–124.
- [2] H. H. Lund, T. Klitbo, and C. Jessen, "Playware technology for physically activating play," *Artifical Life and Robotics Journal*, vol. 9, no. 4, pp. 165–174, 2005.
- [3] T. W. Malone, "What makes computer games fun?" *Byte*, vol. 6, pp. 258–277, 1981.
- [4] N. Postman, *The Disappearance of Childhood*. London: Allen, 1983.
- [5] S. Kline, *Out of the Garden: Toys and Children's Culture in the Age of Marketing*. Verso, 1993.
- [6] G. N. Yannakakis, H. H. Lund, and J. Hallam, "Modeling Children's Entertainment in the Playware Playground," in *Proceedings of the IEEE Symposium on Computational Intelligence and Games*. Reno, USA: IEEE, May 2006, pp. 134–141.
- [7] P. Devijver and J. Kittler, *Pattern Recognition - A Statistical Approach*. Engelwood cliffs, NJ: Prentice-Hall, 1982.
- [8] M. Csikszentmihalyi, *Flow: The Psychology of Optimal Experience*. New York: Harper & Row, 1990.
- [9] P. Sweetser and P. Wyeth, "GameFlow: A Model for Evaluating Player Enjoyment in Games," *ACM Computers in Entertainment*, vol. 3, no. 3, July 2005.
- [10] N. Lazzaro, "Why we play games: Four keys to more emotion without story," XEO Design Inc., Technical Report, 2004.
- [11] R. Koster, *A Theory of Fun for Game Design*. Paraglyph Press, 2005.
- [12] J. Read, S. MacFarlane, and C. Cassey, "Endurability, engagement and expectations," in *Proceedings of International Conference for Interaction Design and Children*, 2002.
- [13] P. Vorderer, T. Hartmann, and C. Klimmt, "Explaining the enjoyment of playing video games: the role of competition," in *ICEC conference proceedings 2003: Essays on the future of interactive entertainment*, D. Marinelli, Ed. Pittsburgh: Carnegie Mellon University Press, pp. 107–120.
- [14] D. Choi, H. Kim, and J. Kim, "Toward the construction of fun computer games: Differences in the views of developers and players," *Personal Technologies*, vol. 3, no. 3, pp. 92–104, September 1999.
- [15] H. Iida, N. Takeshita, and J. Yoshimura, "A metric for entertainment of boardgames: its implication for evolution of chess variants," in *IWEC2002 Proceedings*, R. Nakatsu and J. Hoshino, Eds. Kluwer, 2003, pp. 65–72.
- [16] G. N. Yannakakis and J. Hallam, "Evolving Opponents for Interesting Interactive Computer Games," in *From Animals to Animats 8: Proceedings of the 8th International Conference on Simulation of Adaptive Behavior (SAB-04)*, S. Schaal, A. Ijspeert, A. Billard, S. Vijayakumar, J. Hallam, and J.-A. Meyer, Eds. Santa Monica, LA, CA: The MIT Press, July 2004, pp. 499–508.
- [17] ———, "A Generic Approach for Obtaining Higher Entertainment in Predator/Prey Computer Games," *Journal of Game Development*, vol. 1, no. 3, pp. 23–50, December 2005.
- [18] G. N. Yannakakis, "AI in Computer Games: Generating Interesting Interactive Opponents by the use of Evolutionary Computation," Ph.D. thesis, University of Edinburgh, November 2005.
- [19] G. N. Yannakakis and J. Hallam, "Towards Capturing and Enhancing Entertainment in Computer Games," in *Proceedings of the 4th Hellenic Conference on Artificial Intelligence, Lecture Notes in Artificial Intelligence*, vol. 3955. Heraklion, Greece: Springer-Verlag, May 2006, pp. 432–442.
- [20] G. N. Yannakakis, J. Hallam, and H. H. Lund, "Capturing Entertainment through Heart-rate Dynamics in the Playware Playground," in *Proceedings of the 5th International Conference on Entertainment Computing, Lecture Notes in Computer Science*, vol. 4161. Cambridge, UK: Springer-Verlag, 2006, pp. 314–317.
- [21] P. Rani, N. Sarkar, and C. Liu, "Maintaining optimal challenge in computer games through real-time physiological feedback," in *Proceedings of the 11th International Conference on Human Computer Interaction*, 2005.
- [22] S. McQuiggan, S. Lee, and J. Lester, "Predicting User Physiological Response for Interactive Environments: An Inductive Approach," in *Proceedings of the 2nd Artificial Intelligence for Interactive Digital Entertainment Conference*, 2006, pp. 60–65.
- [23] G. Andrade, G. Ramalho, H. Santana, and V. Corruble, "Extending reinforcement learning to provide dynamic game balancing," in *Proceedings of the Workshop on Reasoning, Representation, and Learning in Computer Games, 19th International Joint Conference on Artificial Intelligence (IJCAI)*, August 2005, pp. 7–12.
- [24] M. A. Verma and P. W. McOwan, "An adaptive methodology for synthesising Mobile Phone Games using Genetic Algorithms," in *Congress on Evolutionary Computation (CEC-05)*, Edinburgh, UK, September 2005, pp. 528–535.
- [25] R. Hunnicke and V. Chapman, "AI for Dynamic Difficulty Adjustment in Games," in *Proceedings of the Challenges in Game AI Workshop, 19th Nineteenth National Conference on Artificial Intelligence (AAAI'04)*, 2004.
- [26] P. Spronck, I. Sprinkhuizen-Kuyper, and E. Postma, "Difficulty Scaling of Game AI," in *Proceedings of the 5th International Conference on Intelligent Games and Simulation (GAME-ON 2004)*, 2004, pp. 33–37.
- [27] R. Pfeifer and C. Scheier, *Understanding Intelligence*. Cambridge, MIT Press, 1999.
- [28] G. N. Yannakakis, J. Hallam, and H. H. Lund, "Comparative Fun Analysis in the Innovative Playware Game Platform," in *Proceedings of the 1st World Conference for Fun 'n Games*, 2006, pp. 64–70.
- [29] R. L. Mandryk, K. M. Inkpen, and T. W. Calvert, "Using Psychophysiological Techniques to Measure User Experience with Entertainment Technologies," *Behaviour and Information Technology (Special Issue on User Experience)*, vol. 25, no. 2, pp. 141–158, 2006.
- [30] C. Beal, J. Beck, D. Westbrook, M. Atkin, and P. Cohen, "Intelligent modelling of the user in interactive entertainment," in *Proceedings of the AAAI Spring Symposium on Artificial Intelligence and Interactive Entertainment*, Stanford, 2002, pp. 8–12.
- [31] X. Yao, "Evolving artificial neural networks," in *Proceedings of the IEEE*, vol. 87, no. 9, 1999, pp. 1423–1447.
- [32] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: University of Michigan Press, 1975.
- [33] D. J. Montana and L. D. Davis, "Training feedforward neural networks using genetic algorithms," in *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89)*. San Mateo, CA: Morgan Kaufmann, 1989, pp. 762–767.
- [34] E. Haapalainen, P. Laurinen, P. Junno, H. Tuovinen, and J. Roening, "Methods for classifying spot welding process: A comparative study of performance," in *IEA/AIE*, 2005, pp. 412–421.

# Modeling Player Experience in Super Mario Bros

Chris Pedersen, Julian Togelius and Georgios N. Yannakakis

**Abstract**— This paper investigates the relationship between level design parameters of platform games, individual playing characteristics and player experience. The investigated design parameters relate to the placement and sizes of gaps in the level and the existence of direction changes; components of player experience include fun, frustration and challenge. A neural network model that maps between level design parameters, playing behavior characteristics and player reported emotions is trained using evolutionary preference learning and data from 480 platform game sessions. Results show that challenge and frustration can be predicted with a high accuracy (77.77% and 88.66% respectively) via a simple single-neuron model whereas model accuracy for fun (69.18%) suggests the use of more complex non-linear approximators for this emotion. The paper concludes with a discussion on how the obtained models can be utilized to automatically generate game levels which will enhance player experience.

**Keywords:** Platform games, player satisfaction, content creation, fun, preference learning, entertainment modeling, neuroevolution

## I. INTRODUCTION

Numerous theories exist regarding what makes computer games fun, as well as which aspects contribute to other types of player experience such as challenge, frustration and immersion [1], [2], [3], [4], [5]. These theories have originated in different research fields and in many cases independently of each other (however, there is substantial agreement on several counts, e.g. regarding the importance of challenge and learnability for making a game fun). While useful high-level guidance for game design, none of these theories is quantitative — derived models of player experience are not mathematical expressions — and they tend to apply to games in general rather than specific aspects of specific games. This means that if we want to develop algorithms that design or adapt games (or aspects of games) automatically, we have to make several auxiliary assumptions in order to achieve the necessary specificity and preciseness of our models.

It seems clear that we need empirical research on particular games to acquire such models. Recently, research in player satisfaction modeling has focused on empirically measuring the effects on player experience of changing various aspects of computer games, such as non-player character (NPC) playing styles in the Pac-Man game [6]. Similarly, efficient quantitative models of player satisfaction have been constructed using in-game data, questionnaires and physiological measurements in augmented-reality games [7].

At the same time, a parallel research direction aims to find methods for automatically generating entertaining game

content. Automatic content generation is likely to be of great importance to computer game development in the future; both offline, for making the game development process more efficient (design of content such as environments and animations now consume a major part of the development budget for most commercial games) and online, for enabling new types of games based on player-adapted content. These efforts see some aspect of a game as variable, define a fitness (“goodness”) function based on a theory of player satisfaction, and use a learning or optimization algorithm to change the variable aspect of the game so as to become more “fun” according to some definition. The literature on this is so far scarce, as it is a new research direction. The aspects of games that have been optimized include:

- Environments, such as tracks for racing games [8] and levels for platform games [9].
- Narrative [10]
- Rules for board games [11], [12] and Pac-Man-like games [13].

What the above studies have in common is that the fitness or cost functions used for optimization have been somewhat arbitrary, in that they have been based on intuition in combination with some qualitative theory of player experience. Optimization of game aspects based on empirically derived models have so far been limited to the impact of NPC behavior [6] and the adjustment of NPC behavioral parameters for maximizing satisfaction in games [14]. To the best of our knowledge, game content such as rules or environments has not been generated based on empirically derived models. We consider better modeling of player experience to be of utmost importance for making automatic content generation techniques more sophisticated and usable.

The work we describe in this paper is concerned with the construction of computational models of player experience, derived from gameplay interaction, which can be used as fitness functions for game content generation. We use a modified version of a classic platform game for our experiments and collect player data through the Internet.

In the following, we describe the game used for our experiments; which player interaction data was collected and how; the preference learning method we used to construct player experience models; how feature selection was used to reduce the number of features used in the model; results of an initial statistical analysis; and the results of training nonlinear perceptrons to approximate the functions mapping between selected gameplay features and aspects of player experience. Finally, we discuss how the induced models will be used for automatically generating game content. This paper significantly extends [15] in which only the core ideas

The authors are with the Center for Computer Games Research, IT University of Copenhagen, Rued Langgaards Vej 7, DK-2300 Copenhagen S, Denmark. Emails: gammabyte@gmail.com, {juto, yannakakis}@itu.dk

of the methodology proposed are outlined.

## II. TESTBED PLATFORM GAME

The test-bed platform game used for our studies is a modified version of Markus Persson's *Infinite Mario Bros* (see Fig. 1) which is a public domain clone of Nintendo's classic platform game *Super Mario Bros*. The original Infinite Mario Bros is playable on the web, where Java source code is also available<sup>1</sup>.

The gameplay in Super Mario Bros consists of moving the player-controlled character, Mario, through two-dimensional levels, which are viewed sideways. Mario can walk and run to the right and left, jump, and (depending on which state he is in) shoot fireballs. Gravity acts on Mario, making it necessary to jump over holes (or gaps) to get past them. Mario can be in one of three states: Small (at the beginning of a game), Big (can crush some objects by jumping into them from below), and Fire (can shoot fireballs).

The main goal of each level is to get to the end of the level, which means traversing it from left to right. Auxiliary goals include collecting as many as possible of the coins that are scattered around the level, clearing the level as fast as possible, and collecting the highest score, which in part depends on the number of collected coins and killed enemies.

The presence of gaps and moving enemies are the main challenges of Mario. If Mario falls down a gap, he loses a life. If he touches an enemy, he gets hurt; this means losing a life if he is currently in the Small state, whereas if he is in the Big and Fire state he shifts to Small and Large state respectively. However, if he jumps so that he lands on the enemy from above, the outcome is dependent on the enemy: Most enemies (e.g. goombas, fireballs) die from this treatment; others (e.g. piranha plants) are not vulnerable to this and proceed to hurt Mario; finally, turtles withdraw into their shells if jumped on, and these shells can then be picked up by Mario and thrown at other enemies to kill them.

Certain items are scattered around the levels, either out in the open, or hidden inside blocks of brick and only appearing when Mario jumps at these blocks from below so that he smashes his head into them. Available items include coins which can be collected for score and for extra lives (every 100 coins), mushrooms which make Mario grow Big if he is currently Small, and flowers which make Mario turn into the Fire state if he is already Big.

No textual description can fully convey the gameplay of a particular game. Only some of the main rules and elements of Super Mario Bros are explained above; the original game is one of the world's best selling games, and still very playable more than two decades after its release in the mid-eighties. Its game design has been enormously influential and inspired countless other games, making it a good experiment platform for player experience modeling.

While implementing most features of Super Mario Bros, the standout feature of Infinite Mario Bros is the automatic generation of levels. Every time a new game is started,



Fig. 1. Test-bed game screenshot, showing Small Mario jumping over a piece of flat terrain surrounded by two gaps.

levels are randomly generated by traversing a fixed width and adding features (such as blocks, gaps and opponents) according to certain heuristics. In our modified version of Infinite Mario Bros most of the randomized placement of level features is fixed since we concentrate on a few selected game level parameters that affect game experience.

## III. DATA COLLECTION

Before any modeling could take place we needed to collect data to train the model on. We collected three types of data from hundreds of players over the Internet:

- 1) Controllable features of the game, i.e. the parameters used for level generation, and affecting the type and difficulty of the level. These were varied systematically to make sure all variants of the game were compared.
- 2) Gameplay characteristics, i.e. how the user plays the game. We measured statistical features such as how often and when the player jumped, ran, died etc. These features cannot be directly controlled by the game as they depend solely on the player's skill and playing style in a particular game level.
- 3) The player's experience of playing the game, measured through a 4-alternative forced choice questionnaire administered after playing two pairs of games with different controllable features and asking the players to rank the games in order of emotional preference.

Below we describe in detail which features were collected for each type of data.

### A. Controllable features

We modified the existing level generator to create levels according to four controllable parameters presented below. Three of these parameters are dedicated to the number, width and placement of gaps. The fourth parameter turns a new function, the direction switch, on or off.

- The number of gaps in the level,  $G$ .
- The average width of gaps,  $E\{G_w\}$ .
- The spatial diversity of gaps which is measured by the entropy of the number of gaps appearing in a number of

<sup>1</sup><http://www.mojang.com/notch/mario/>

$G$  (equally-spaced) segments of the level. The entropy of gap-placements  $H_g$  in the  $G$  segments is calculated and normalized into  $[0, 1]$  via (1):

$$H_g = \left[ -\frac{1}{\log G} \sum_{i=1}^G \frac{g_i}{G} \log \left( \frac{g_i}{G} \right) \right] \quad (1)$$

where  $g_i$  is the number of gap-placements into level segment  $i$ . If the gaps are placed in all  $G$  level segments uniformly then  $g_i = 1$  for all  $G$  parts and  $H_g$  will be 1; if all gaps are placed in one level segment  $H_g$  is zero. This controllable feature provides a notion of unpredictability of gaps and, therefore, jumps in the level. Unpredictability has been identified as an important factor of playing experience [16].

- Number of direction switches,  $S$ . No direction switch means that the player needs to move from left to right in order to complete the level, as in the original Super Mario Bros. If one or more direction switch is present, the level direction will be mirrored at certain points, forcing the player to turn around and go the other way, until reaching the end of the level or the next switch.

The selection of these particular controllable features was done after consulting game design experts, and with the intent to find features which were common to most, if not all, platform games.

Two states (low and high) for each of the four controllable parameters above are investigated. The combinations of these states result in  $2^4 = 16$  different variants of the game which are used in the user study designed. In the Super Mario Bros game investigated here the number of coins, opponents, coin blocks, powerup blocks and empty blocks are fixed to 15, 3, 4, 2, and 8 respectively.

#### B. Gameplay features

Several statistical features are extracted from playing data which are logged during gameplay and include game completion time, time spent on various tasks (e.g. jumping, running), information on collected items (e.g. type and amount), killed enemies (e.g. type, amount, way of killing) and information on how the player died. The choice of those specific statistical features is made in order to cover a decent amount of Super Mario Bros playing behavior dynamics. In addition to the four controllable game features that are used to generate Super Mario Bros levels presented earlier, the following statistical features are extracted from the gameplay data collected and are classified in five categories: jump, time, item, death, kill and misc.

Jump: difference between the total number of jumps,  $J$ , minus the number of gaps,  $G$ ; number of jumps over gaps or without any purpose (e.g. to collect an item, to kill an opponent),  $J'$ ; difference between  $J'$  and the number of gaps,  $G$ ; and a jump difficulty heuristic,  $J_d$ , which is proportional to the number of Super Mario deaths due to gaps, number of gaps and average gap width.

Time: completion time,  $t_c$ ; playing duration of last life over total time spent on the level,  $t_{ll}$ ; percentage of time that the player: is standing still,  $t_s$ , running,  $t_r$ , is on large Mario mode,  $t_l$ , is on fire Mario mode,  $t_f$ , is on powerup mode,  $t_p$ , is moving left,  $t_L$ , is moving right,  $t_R$ , and is jumping,  $t_j$ .

Item: number of collected items (coins, destroyed blocks and powerups) over total items existent in the level,  $n_I$ ; number of times the player kicked an opponent shell,  $n_s$ ; number of coins collected over the total number of coins existent in the level,  $n_c$ ; number of empty blocks destroyed over the total number of empty blocks existent in the level,  $n_{eb}$ ; number of coin blocks pressed over the total number of coin blocks existent in the level,  $n_{cb}$ ; number of powerup blocks pressed over the total number of powerup blocks existent in the level,  $n_p$ ; and the sum of all blocks pressed or destroyed over the total number of blocks existent in the level  $n_b = n_{eb} + n_{cb} + n_p$ .

Death: number of times the player was killed: by an opponent,  $d_o$ ; by jumping into a gap,  $d_g$ ; by jumping into a gap over the total number of deaths  $d_j$ .

Kill: number of opponents died from stomping over the total number of kills,  $k_s$ ; number of opponents died from fire-shots over the total number of kills,  $k_f$ ; total number of kills over total number of opponents,  $k_T$ ; number of opponent kills minus number of deaths caused by opponents,  $k_P$ ; and number of cannonballs killed,  $k_c$ .

Misc: number of times the player shifted the mode (Small, Big, Fire),  $n_m$ ; number of times the run button was pressed,  $n_r$ ; number of ducks,  $n_d$ ; number of cannonballs spawned,  $n_{cs}$ ; and whether the level was completed or not  $C$  (boolean).

#### C. Reported player experience and experimental protocol

We designed a game survey study to solicit pairwise emotional preferences of subjects playing different variants of the test-bed game by following the experimental protocol proposed in [7]. Each subject plays a predefined set of four games in pairs: a game pair of game  $A$  and game  $B$  played in both orders. The games played differ in the levels of one or more of the four controllable features presented previously. For each completed pair of games  $A$  and  $B$ , subjects report their emotional preference using a 4-alternative forced choice (4-AFC) protocol:

- game  $A$  [ $B$ ] was/felt more  $E$  than game  $B$  [ $A$ ] game (cf. 2-alternative forced choice);
- both games were/felt equally  $E$  or
- neither of the two games was/felt  $E$ .

Where  $E$  is the emotional state under investigation and contains *fun, challenging, boring, frustrating, predictable* and *anxious*. The selection of these six states is based on their relevance to computer game playing and their popularity when it comes to game-related user studies [17]. In this initial investigation of player experience we focus only on three emotions: *fun, challenge* and *frustration*.

Data is collected over the Internet. Users are recruited via posts on blogs and mailing lists and directed to a web page containing a Java applet implementing the game and questionnaire<sup>2</sup>. As soon as the four games are played and the questionnaire is completed, all the features (controllable, gameplay and player experience) are saved in a database at the server hosting the website and applet. Data collection is still in progress and at the moment of writing, 181 subjects have participated in the survey experiment. The minimum number of experiment participants required is determined by  $C_2^{16} = 120$ , this being the number of all combinations of 2 out of 16 game variants. The experimental protocol is designed in such a way that at least 2 preference instances should be obtained for each pair of the 16 game variants played in both orders (1 preference instance per playing order). The analysis presented in this paper is based on the 240 game pairs (480 game sessions) played by the first 120 subject participants.

#### IV. PREFERENCE LEARNING FOR MODELING PLAYER EXPERIENCE

Based on the data collected in the process described above, we try to approximate the function from gameplay features (e.g. number of coins gathered) and controllable game level features (e.g. number of gaps) to reported emotional preferences using neuroevolutionary preference learning.

The data is assumed to be a very noisy representation of the underlying function, given the high level of subjectivity of human preferences and the expected variation in playing styles. Together with the limited amount of training data, this makes overfitting a potential hazard and mandates that we use a robust function approximator. We believe that a non-linear function such as an artificial neural network (ANN) is a good choice for approximating the mapping between reported emotions and input data. Thus, a simple single-neuron (perceptron) is utilized for learning the relation between features (ANN inputs) — selected from feature selection schemes presented in Section V — and the value of the investigated emotional preference (ANN output) of a game. The main motivation for using a single neuron instead of a multi-layered perceptron (MLP) in this study is that we want to be able to analyze the trained function approximator. While an MLP can potentially approximate the function investigated with a higher accuracy, it is much easier for a human to understand the obtained function when represented as a single-neuron ANN.

The single neuron uses the sigmoid (logistic) activation function; connection weights take values from -5 to 5 to match the normalized input values that lie in the [0, 1] interval. Since there are no prescribed target outputs for the learning problem (i.e. no differentiable output error function), ANN training algorithms such as back-propagation are inapplicable. Learning is achieved through artificial evolution by following the preference learning approach presented in

<sup>2</sup>The game and questionnaire are available at [www.bluenight.dk/mario.php](http://www.bluenight.dk/mario.php)

[18]. A generational genetic algorithm (GA) is implemented, using a fitness function that measures the difference between the subject's reported emotional preferences and the relative magnitude of the corresponding model (ANN) output.

#### V. FEATURE SELECTION

We would like our model to be dependent on as few features as possible, both to make it easier to analyze, and to make it more useful for incorporation into future games for purposes of e.g. content creation. Therefore, feature selection is utilized to find the feature subset that yields that most accurate user model and save computational effort of exhaustive search on all possible feature combinations. The quality of the predictive model constructed by the preference learning outlined above depends critically on the set of input data features chosen. Using the extracted features described earlier the *n best individual feature selection* (nBest), the *Sequential Forward Selection* (SFS) and the *Perceptron Feature Selection* (PFS) schemes are applied and compared.

##### A. nBest

nBest feature selection ranks the features used individually in order of model performance; the chosen feature set of size n is then the first n features in this ranking. The nBest method is used for comparative purposes, being the most popular technique for feature selection.

##### B. SFS

SFS is a bottom-up search procedure where one feature is added at a time to the current feature set. The feature to be added is selected from the subset of the remaining features such that the new feature set generates the maximum value of the performance function over all candidate features for addition. The SFS method has been successfully applied to a wide variety of feature selection problems, yielding high performance values with minimal feature subsets [7], [19]

##### C. PFS

The third method we investigate is Rosenblatt's perceptron as a methodology for selecting appropriate feature subsets. Our algorithm which is similar to [20] is adjusted to match preference learning problems. Thus, the perceptron used employs the sigmoid activation function in a single output neuron. The ANN's initial input vector has the size of the number of features examined. The perceptron feature selection (PFS) procedure is as follows:

**Step 1** Use artificial evolution to train the perceptron on the pairwise preferences (see Section IV). Performance of the perceptron is evaluated through 3-fold cross-validation. The initial input vector consists of all features extracted  $\mathcal{F}$  (40 in this paper).

**Step 2** Eliminate all features  $\mathcal{F}'$  whose corresponding absolute connection weight values are smaller than  $E\{|w|\} - \sigma\{|w|\}$ , where w is the connection weight vector.

**Step 3** If  $\mathcal{F}' = \emptyset$  continue to Step 4, otherwise use the remaining features and go to Step 1.

**Step 4** Evaluate all feature subsets obtained using the neuro-evolution preference learning approach presented in Section IV.

Note that all three methods are incomplete. Neither is guaranteed to find the optimal feature set since neither searches all possible combinations (they are all variants of hill-climbing). To evaluate the performance of each input feature subset, the available data is randomly divided into thirds and training and validation data sets consisting of 2/3 and 1/3 of the data respectively are assembled. The performance of each user model is measured through the average classification accuracy of the model in three independent runs using the 3-fold cross-validation technique on the three possible independent training and validation data sets. Since we are interested in the minimal feature subset that yields the highest performance we terminate the SFS selection procedure when an added feature yields equal or lower validation performance to the performance obtained without it. On the same basis, we store all feature subsets selected by PFS and explore the highest performing subset starting with the smallest feature subset generated.

## VI. STATISTICAL ANALYSIS

This section describes testing for correlations between playing order, controlled features and gameplay features and the reported emotions of fun, challenge and frustration.

To check whether the order of playing Super Mario game variants affects the user's judgement of emotional preferences, we follow the order testing procedure described in [6] which is based on the number of times that the subject prefers the first or the second game in both pairs. The statistical analysis shows that order of play does not affect the emotional preferences of fun and frustration; however a statistically significant effect is observed in challenge preferences ( $p\text{-value} = 0.006$ ). The effect reveals a preference for the second game played which implies the existence of random noise in challenge preference expression. On the other hand, the insignificant order effects of fun and frustration, in part, demonstrate that effects such as a user's possible preference for the very first game played and the interplay between reported emotions and familiarity with the game are statistically insignificant.

More importantly, we performed an analysis for exploring statistically significant correlations between subject's expressed preferences and extracted features. Correlation coefficients are obtained through  $c(\mathbf{z}) = \sum_{i=1}^{N_s} \{z_i/N_s\}$ , where  $N_s$  is the total number of game pairs where subjects expressed a *clear preference* for one of the two games (e.g.  $A \succ B$  or  $A \prec B$ ) and  $z_i = 1$ , if the subject preferred the game with the larger value of the examined feature and  $z_i = -1$ , if the subject chooses the other game in the game pair  $i$ . Note that,  $N_s$  is 161, 189 and 151 respectively, for reported fun, challenge and frustration.

The variation of the  $N_s$  numbers above indicates, in part, the difficulty in expressing a clear emotional preference on different game variants. The percentage of  $A \succ B$

and  $A \prec B$  selection occurrences over all 240 preference instances for different emotional states varies from 78.7% (challenge) to 62.9% (frustration). These percentages provide some first evidence that the selected game level and rule parameters have an dissimilar impact on the emotional states investigated. For instance, challenge appears to be very much affected by varying the selected parameters whereas frustration, on the contrary, does not appear as an emotion which is directly affected by variations in the game.

TABLE I  
TOP TEN STATISTICALLY SIGNIFICANT ( $P\text{-VALUE} < 1\%$ ) CORRELATION COEFFICIENTS BETWEEN REPORTED EMOTIONS AND EXTRACTED FEATURES.

|          | Fun   |            | Challenge |          | Frustration |
|----------|-------|------------|-----------|----------|-------------|
| $n_s$    | 0.345 | $C$        | -0.600    | $C$      | -0.826      |
| $n_{cb}$ | 0.311 | $n_p$      | -0.480    | $n_p$    | -0.815      |
| $k_T$    | 0.256 | $d_j$      | 0.469     | $n_{cb}$ | -0.688      |
| $n_r$    | 0.253 | $d_g$      | 0.447     | $d_g$    | 0.578       |
| $t_L$    | 0.237 | $J_d$      | 0.439     | $d_j$    | 0.564       |
| $k_P$    | 0.222 | $E\{G_w\}$ | 0.409     | $n_I$    | -0.544      |
| $t_r$    | 0.192 | $n_d$      | -0.368    | $t_s$    | 0.520       |
|          |       | $t_{ll}$   | -0.312    | $k_f$    | -0.515      |
|          |       | $n_{cb}$   | -0.292    | $t_{ll}$ | -0.513      |
|          |       | $G$        | -0.287    | $n_c$    | -0.511      |

### A. Fun

Statistically significant correlations are observed between reported fun and seven features: number of times the player kicked a turtle shell, proportion of coin blocks that were "pressed" (jumped at from below), proportion of opponents that were killed, number of times the run button was pressed, proportion of time spent moving left, number of enemies killed minus times died, and proportion of time spent running. All of these were positive correlations.

Such correlations draw a picture of most players enjoying a fast paced game that includes near-constant progress, plenty of running, many enemies killed and many coins collected from bouncing off the coin blocks. One might argue that this picture fits with the concept of *Flow*, in that the player makes unhindered progress [3]. However, the Flow concept also includes a certain level of challenge, and no features that signify challenge are associated with fun in this case. It might be that players enjoy when the game is easy — at least when they only play a single level of the game.

The feature that correlates the most with fun preferences is kicking turtle shells. Kicking a turtle shell is a simple action which often results in the unfolding of a relatively complex sequence of events, as the shell might bounce off walls, kill enemies, fall into gaps etc. The fun inherent in setting of complex chains of events with simple actions is something many players can relate to and which features prominently in many games, but which is to our knowledge not part of any of the "established" theories of what makes games fun.

### B. Challenge

Eighteen features are significantly correlated with challenge. The ten most highly correlated are (+/- in parenthesis signifies positive or negative correlation): whether the level was completed (-), proportion of power-up blocks pressed (-), proportion of Mario deaths that were due to falling into a gap (+), number of times Mario died from falling into a gap (+), jump difficulty (+), average width of gaps (+), number of times Mario ducked (-), proportion of time spent in the last life (-), proportion of coin blocks that were pressed (-), and the number of gaps (-). In addition, a weaker but still significant positive correlation was found between gap entropy,  $H_g$ , and challenge.

A first observation is that it is obviously much easier to predict challenge than to predict fun — many more features are significantly correlated, and the correlations are stronger. It also seems that challenge is somehow orthogonal to fun, as almost none of the features that are correlated with challenge are correlated with fun. The exception is the proportion of coin blocks pressed, but while this feature is positively correlated with fun it is negatively correlated with challenge. (This is somewhat expected: if the level is so hard that the player has to struggle to survive it, she does not have time to make detours in order to collect more coins.)

Most of the correlations are easy to explain. That a level is perceived as less challenging if you complete it should not come as a surprise to anyone. Likewise, we can understand that players think a level is hard when they repeatedly die from falling into gaps. Three particularly interesting correlations are those between the controllable features and challenge: increase in gap width,  $E\{G_w\}$ , and gap entropy,  $H_g$ , imply increased challenge whereas increased number of gaps,  $G$ , implies a linear decrease of challenge. These effects suggest that challenge can be controlled to a degree by changing the number, width and distribution of gaps.

The correlation with number of ducks would have been easy to explain — if it was positive. The main reason for ducking in Super Mario Bros (at least in the tested levels) is to avoid cannonballs. To the authors, cannons are perceived as some of the most difficult elements on a level. However, players reported lower challenge on levels where they ducked many times. We have yet to find an explanation for this.

### C. Frustration

Twenty-eight features are significantly correlated with frustration, and some of the correlations are extremely strong. Of the top ten correlated features, most are also in the top ten list for features correlated with challenge, and correlated in the same way. The exceptions are proportion of collected items (-), time spent standing still (+), proportion of killed opponents that were killed with fireballs (-), and proportion of coins collected (-).

From these new features, it seems that a frustrated player is most likely one that spends time standing still and thinking about how to overcome the next obstacle; is far too busy overcoming obstacles to collect coins and power-ups; and

as a result of not collecting power-ups is rarely in the Fire Mario state (necessary to shoot fireballs). But frustration can also be very well predicted from not winning the level and from falling into gaps often, just like challenge.

### D. Controllable features and intra-correlations

When only looking at linear correlations, it would appear that fun is not connected to any of the four controllable features. Fun is also less strongly correlated with gameplay features than is the case for challenge and frustration. The latter two emotions are easier to model with linear models, and are also strongly correlated with controllable features, namely gap entropy, gap width and number of gaps.

The three emotions are also all significantly ( $p$ -value  $< 0.001$ ) correlated to each other. The correlation coefficient  $c(\mathbf{z})$  between challenge and fun and between challenge and frustration is 0.346 and 0.462, respectively, while the corresponding correlation between fun and frustration is -0.284. The positive effect of challenge on fun and frustration, combined with the negative effect of fun on frustration indicate the nonlinearity (and possibly complexity) of those emotions' interrelationships.

## VII. NONLINEAR PREFERENCE MODELLING

The correlations calculated above provide linear relationships between individual features and reported emotions. However, these relationships are most likely more complex than can be captured by linear models. The aim of the analysis presented below is to construct non-linear computational models for reported emotions and analyze the relationship between the selected features and expressed preferences.

For this purpose we evolve weights for nonlinear perceptrons as described in Section IV. The weights of the highest performing networks are presented in Table II. All evolved networks performed much better than networks with random weights, which reached chance level prediction accuracy.

TABLE II  
LEARNING FROM PREFERENCES: FEATURES AND CORRESPONDING CONNECTION WEIGHTS FOR HIGHEST PERFORMING ANNS

|       | Fun    | Challenge | Frustration |
|-------|--------|-----------|-------------|
| $t_L$ | 4.905  | $t_s$     | -1.703      |
| $k_s$ | 0.942  | $J_d$     | 3.805       |
| $L$   | -3.873 | $n_{eb}$  | -1.502      |
|       |        | $k_c$     | 1.073       |
|       |        | $k_s$     | -0.189      |
|       |        | $t_{ll}$  | -1.851      |
|       |        | $J_d$     | -0.995      |
|       |        | $d_g$     | 0.233       |

### A. Fun

In the comparison between the three different selection mechanisms applied it is evident that SFS has advantages over nBest and PFS for fun preferences (see Fig. 2(a)). nBest achieves a satisfactory performance (67.92%) but requires 10 features as inputs to the ANN. PFS generates the lowest classification accuracies; its best network has an accuracy of 63.52% with a selected subset of 11 input features.

The best obtained perceptron model of fun preferences is designed by SFS. This model achieves a performance

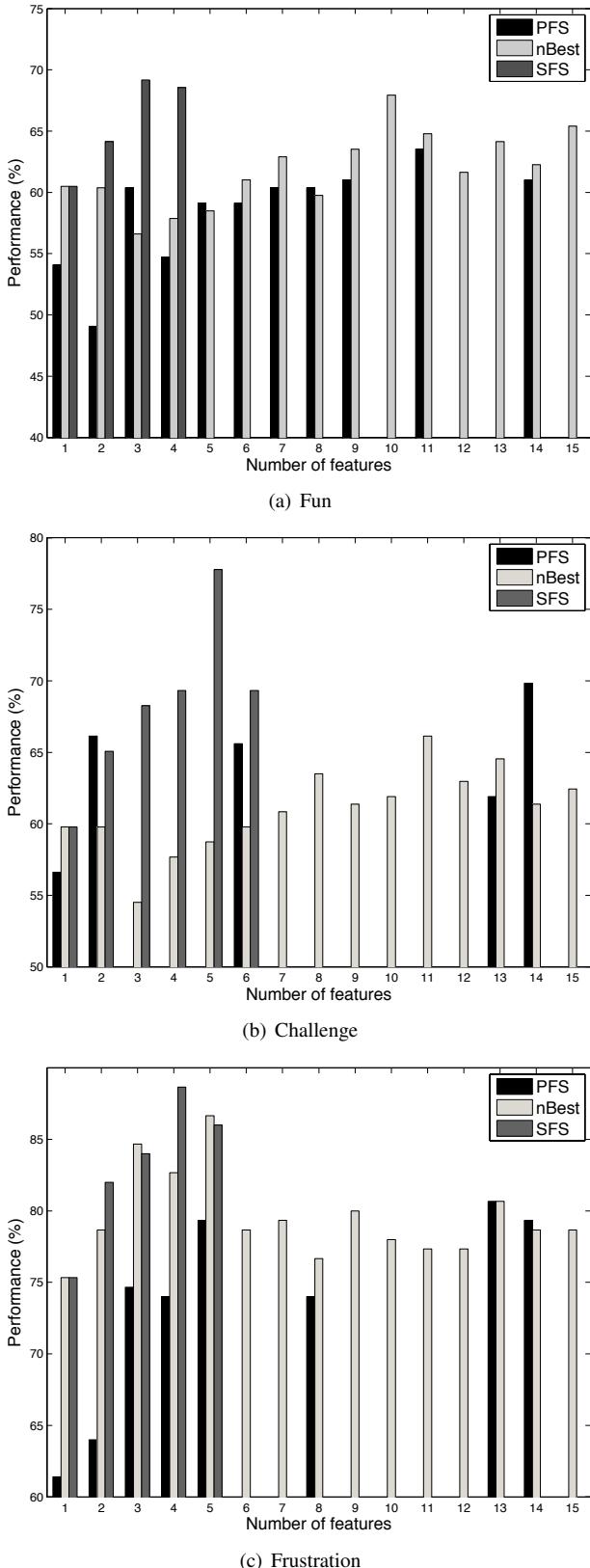


Fig. 2. Performance comparison of feature selection mechanisms on emotional preferences.

of 69.18% which is with a selected feature subset of size three. The selected perceptron input vector consists of the time spent moving left  $t_L$ , the number of opponents died from stomping over the total number of kills,  $k_s$ , and the controllable *switching* feature which is defined as the percentage of level played in the left direction  $L^3$ .

Fun is the least correlated of the three modeled emotions, and the hardest to model with a nonlinear perceptron as well. Still, it's remarkable that this complex emotion can be predicted to a moderate degree simply by observing that Mario keeps running left and kills enemies by stomping.

### B. Challenge

The best-performing ANN for challenge prediction has an accuracy of 77.77%. It is more complex than the best fun predictor, using five features: time spent standing still (-), jump difficulty (+), proportion of coin blocks pressed (-), number of cannonballs killed (-) and proportion of kills by stomping (-). While the jump difficulty heuristic has the largest corresponding weight — a testament to the central role of gap placement and size for challenge — it is also the only feature related to gaps used by this model, pointing to the adequateness of this particular heuristic.

### C. Frustration

Our best evolved ANN for predicting frustration has an accuracy of 88.66%. We can predict with near-certainty whether the player is frustrated by the current game by just calculating the time spent standing still (+), the proportion of time spent on last life (-), the jump difficulty (-), and the proportion of deaths due to falling in gaps (+).

Somewhat surprisingly, time spent standing still counts *against* challenge, whereas it is a strong *positive* predictor of frustration. This observation could be valuable if trying to design a feedback system that keeps the game challenging but not frustrating. Another feature that has different effect on challenge and frustration is jump difficulty, where frustration is connected with *lower* jump difficulty. Maybe the player gets frustrated by falling into gaps that she knows are not that hard.

That the player feels frustrated when dying after a short time during his last life is understandable — many players feel that their last attempt should be their best. Additionally, a high frustration level can cause the player to care less about the game and play worse in her final life.

## VIII. DISCUSSION

While we have found relatively good predictors for all three emotions, two problems remain: the predictions (at least for fun and challenge) are still not as good as we would like them to be, and we cannot reliably predict fun from controllable features. As controllable features (such as level

<sup>3</sup>The  $L$  feature is there to correct for the fact that when the level direction switches, Mario moves right rather than left to move forward, and so  $t_L$  is diminished. This points to an oversight on our part when designing the gameplay features: we should have measured the time spent moving *towards the end of the level* rather than moving left.

design parameters) are those that we can vary, and therefore those that can be optimized by evolution or other global optimizers, we need to be able to predict emotions at least partly from controllable features.

This points to the need for better models and/or features. First of all, we will try to induce multi-layer perceptron models of emotional preferences. MLPs have the advantage of universal approximation capacity; in particular, combinatorial relationships (such as XOR) can be represented. We might very well have a situation where one controllable feature (such as gap width) can be both negatively and positively connected with an emotion (such as frustration) depending on the player's playing style, as measured through gameplay features (such as number of jumps). Such relationships can be captured by MLPs but not by nonlinear perceptrons.

Data collection is continuing at the time of writing, and probably at the time of reading (the reader is welcome to contribute by visiting the project's web site), the new data will be used to improve the accuracy of our predictions.

Depending on the success of finding predictors partly dependent on controllable features, we might need to design new controllable features or revise the existing ones. New features might include the number and type of enemies, the existence of dead ends in the level (forcing backtracking) etc.

After good models have been learned, evolutionary algorithms will be used to optimize the level design parameters (relating to gaps and switches) for different objectives. We hope to, this way, be able to generate levels that tailor the playing experience according to the needs of the game design (e.g. a challenging level combined with a non-frustrating experience). The success of our optimization attempts will be validated with further user studies.

Another question concerns the generality of the results gathered here — do they apply to just the players and the particular game tested here, or do they have wider applicability? We venture that, as Super Mario Bros more or less defined the platform game genre, the results apply to some extent to all games of the same genre. Further, the population of experimental subjects is believed to be very diverse, but this needs to be verified. A possible critique is that the emotions reported are those that have been elicited after only a few minutes of play. It is possible that challenge or variety (gap entropy) would factor in more if play sessions were longer, so subjects would have had a chance of getting bored with the game.

## IX. CONCLUSIONS

We designed a user study focused on a version of the Super Mario Bros platform game, in which a population of subjects played in a number of different versions (mainly differing in the game environments encountered). Controllable features and emergent gameplay features were correlated with reported emotions during gameplay. We found a large number of statistically significant correlations, and were able to train good predictors of player emotions using preference learning and neuroevolution. These results will be improved upon and form the basis for attempts to automatically generate

environments for this game using artificial evolution with the induced player experience models as fitness functions.

## ACKNOWLEDGMENTS

The authors would like to thank Aki Järvinen and Markus Persson for insightful discussions, and all subjects that participated in the experiments.

## REFERENCES

- [1] C. Bateman and R. Boon, *21st Century Game Design*. Charles River Media, 2005.
- [2] K. Isbister and N. Schaffer, *Game Usability: Advancing the Player Experience*. Morgan Kaufman, 2008.
- [3] M. Csikszentmihalyi, *Flow: the Psychology of Optimal Experience*. Harper Collins, 1990.
- [4] R. Koster, *A theory of fun for game design*. Paraglyph press, 2005.
- [5] J. Juul, *Half-real*. MIT Press, 2005.
- [6] G. N. Yannakakis and J. Hallam, "Towards optimizing entertainment in computer games," *Applied Artificial Intelligence*, vol. 21, pp. 933–971, 2007.
- [7] ———, "Entertainment modeling through physiology in physical play," *International Journal of Human-Computer Studies*, vol. 66, pp. 741–755, 2008.
- [8] J. Togelius, R. De Nardi, and S. M. Lucas, "Towards automatic personalised content creation in racing games," in *Proceedings of the IEEE Symposium on Computational Intelligence and Games*, 2007.
- [9] K. Compton and M. Mateas, "Procedural level design for platform games," in *Proceedings of the Artificial Intelligence and Interactive Digital Entertainment International Conference (AIIDE)*, 2006.
- [10] M. J. Nelson, C. Ashmore, and M. Mateas, "Authoring an interactive narrative with declarative optimization-based drama management," in *Proceedings of the Artificial Intelligence and Interactive Digital Entertainment International Conference (AIIDE)*, 2006.
- [11] C. Browne, "Automatic generation and evaluation of recombination games," Ph.D. dissertation, Queensland University of Technology, Brisbane, Australia, 2008.
- [12] J. Marks and V. Horn, "Automatic design of balanced board games," in *Proceedings of the Artificial Intelligence and Interactive Digital Entertainment International Conference (AIIDE)*, 2007, pp. 25–30.
- [13] J. Togelius and J. Schmidhuber, "An experiment in automatic game design," in *Proceedings of the IEEE Symposium on Computational Intelligence and Games*, 2008.
- [14] G. N. Yannakakis and J. Hallam, "Real-time Game Adaptation for Optimizing Player Satisfaction," *IEEE Transactions on Computational Intelligence and AI in Games*, 2009, (to appear).
- [15] C. Pedersen, J. Togelius, and G. Yannakakis, "Optimization of platform game levels for player experience," in *Proceedings of the Artificial Intelligence and Interactive Digital Entertainment International Conference (AIIDE)*, 2009.
- [16] T. W. Malone, "What makes computer games fun?" *Byte*, vol. 6, pp. 258–277, 1981.
- [17] R. L. Mandryk and M. S. Atkins, "A Fuzzy Physiological Approach for Continuously Modeling Emotion During Interaction with Play Environments," *International Journal of Human-Computer Studies*, vol. 65, pp. 329–347, 2007.
- [18] G. N. Yannakakis and J. Hallam, "Game and Player Feature Selection for Entertainment Capture," in *Proceedings of the IEEE Symposium on Computational Intelligence and Games*. Hawaii, USA: IEEE, April 2007, pp. 244–251.
- [19] G. N. Yannakakis, M. Maragoudakis, and J. Hallam, "Preference Learning for Cognitive Modeling: A Case Study on Entertainment Preferences," *IEEE Systems, Man and Cybernetics; Part A: Systems and Humans*, 2009, (to appear).
- [20] M. Mejia-Lavalle and G. Arroyo-Figueroa, "Power System Database Feature Selection Using a Relaxed Perceptron Paradigm," in *Proceedings of 5th Mexican International Conference on Artificial Intelligence, LNCS*. Springer Berlin/Heidelberg, 2006, pp. 522–531.

## Dynamic Difficulty Adjustment Through an Adaptive AI

Mirna Paula Silva, Victor do Nascimento Silva and Luiz Chaimowicz

*Departamento de Ciéncia da Computaçao*

*Instituto de Ciéncias Exatas*

*Universidade Federal de Minas Gerais, Brazil*

*Email:* {mirnasilva, vnsilva}@ufmg.br, chaimo@dcc.ufmg.br

**Abstract**—Dynamic Difficulty Adjustment (DDA) consists in an alternative to the static game balancing performed in game design. DDA is done during execution, tracking the player’s performance and adjusting the game to present proper challenges to the player. This approach seems appropriate to increase the player entertainment, since it provides balanced challenges, avoiding boredom or frustration during the game-play. This paper presents a mechanism to perform the dynamic difficulty adjustment during a game match. The idea is to dynamically change the game AI, adapting it to the player skills. We implemented three different AIs to match player behaviors: beginner, regular and experienced in the game *Defense of the Ancient (DotA)*, a modification (MOD) of the game Warcraft III. We performed a series of experiments and, after comparing all results, the presented mechanism was able to keep up with the player’s abilities on 85% of all experiments. The remaining 15% failed to suit the player’s need because the adjustment did not occur on the right moment.

**Keywords**-Artificial Intelligence; Digital Games; Dynamic Difficulty Adjustment; Dynamic Difficulty Balance; Entertainment

### I. INTRODUCTION

The game industry is growing at a fast pace, globally generating more revenue than film and music industries [1]. Games are considered a great source of entertainment [2] and, due to that, the industry is increasingly investing more resources in research and development. This allows developers to create realistic graphics, deep narratives and complex artificial intelligence (AI), leading to games even closer to reality [3], [4].

Developing realistic games helps the improvement of players immersion which increases their satisfaction [5]. Although this is a good approach, it is not the only way to make games more attractive. According to Yannakakis [6], the player’s psychological factor makes direct influence to this attractiveness, requiring the game to maintain the player interested on it. An approach to captivate the player into the game experience is to make the challenges directly associated to the player’s skill [7]. However, a game may not suit the expectation of players with different skills. While a player may have a hard time in final levels of a game, there may be another player that cannot win the initial ones. This scenario requires that the game dynamically adjusts itself presenting challenges that suits the needs and skills of each player. This game adjustment can be performed by a

technique called dynamic difficulty adjustment (or dynamic difficulty balancing).

This work aims to present a mechanism to perform the difficulty adjustment dynamically during a game match. To achieve this goal we observed and identified the behavior of three different types of player (beginner, regular and experienced) and developed an artificial intelligence that simulates each one of these. The main idea is to present an opponent that is challenging enough for the player without being too hard. Therefore we established an evaluation process to indicate moments during the game match where the player is increasing/decreasing his/her performance. A mechanism were developed to execute adjustments by changing the difficulty of the selected artificial intelligence and for each of these unbalanced moments, the mechanism analysis if is necessary to perform an adjustment or not. The framework selected to implement this approach is a modification (MOD) of the game WARCRAFT III, called DEFENSE OF THE ANCIENT (DotA). At this game, the player control one specific unit called hero and the main challenge is to destroy a main structure that belongs to enemy.

This paper is organized as follows: in Section 2 we present the related work and background on difficulty balance; Section 3 covers the game DotA used as framework of this work; Section 4 addresses the methodology and the proposed mechanism; Section 5 discusses the performed experiments and the obtained results; and finally, in Section 6 we offer our conclusions and future work.

### II. DIFFICULTY BALANCE

Difficulty balance, or difficulty adjustment, consists on doing modifications to parameters, scenarios and/or game behaviors in order to avoid the player’s frustration when facing the game challenges [7], [8]. According to Mateas [9] and Hunicke [10], it is possible to adjust all game features using the correct algorithms, from storytelling to maps and level layouts, all online. These adjustments allow the game to adapt itself to each player, making he/she entertained throughout the game. To make this possible, Andrade et al. [11] describes that the dynamic difficulty adjustment must attend three basic requirements. First of all, the game must automatically identify the players’ skills and adapt to it as fast as possible. Second, the game must track the

player's improvement and regressions, as the game must keep balance according to the player's skill. At last, the adaptive process must not be explicitly perceived by players, keeping game states coherent to previous ones. However, before applying the dynamic difficulty adjustment, it is necessary to understand the meaning of difficulty.

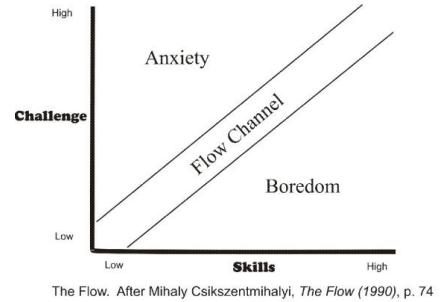
The meaning of difficulty is abstract in many ways and some aspects should be taken into account to evaluate and measure difficulty. For this measuring, we can consider level design characteristics [12], amount of resource or enemies [10], amount of victories or losses [13], among other metrics. Although, dynamic difficulty adjustment is not as simple as just giving player additional health items when in trouble. This problem requires estimation of time and intervention in the right moment, since maintaining the player entertained is a complex task in a interactive context [10].

A wide range of tasks and challenge levels can be found in games. For instance, tasks that require high skill and synchronism (First Person Games), tasks that require logic and problem solving skills (Puzzles), tasks related to planning (Strategy games), and so on [14]. According to Klimmt et al. [14], there is evidence that the completion of tasks and challenge overcoming are directly related to player satisfaction and fun. Yannakakis [15] developed a study about the most popular approaches for player modeling during interaction with entertainment systems. According to this study, most qualitative approaches proposed for player entertainment modeling tends to be based in conceptual definitions proposed by Malone [16] and Csikszentmihalyi [17].

Malone [16] defended the need for a specific motivation during gameplay to entertain the player. The necessary features to reach such motivation are: fantasy, control, challenges and curiosity. The use of fantasy as part of game world could improve player motivation, creating objects, scenarios or situations that the player could explore. Control is a player feeling that he/she is part of game control. Given the interaction of games, all of them makes the player feel involved in game control and the control levels can change from game to game. Challenge proposes that the game should pursue tasks and goals in an adequate level, making the player feel challenged to his/her limits. The uncertainty of completing tasks or goals provided by game mechanics encourages the player motivation. At last, curiosity suggests that game information must be complex and unknown, to encourage exploration and reorganization of information by players. Games must pursue parallel situations or scenarios from the main course since it helps to stimulate the player to explore the unknown [16], [18].

The qualitative approach proposed by Csikszentmihalyi [17] is called flow theory or flow model. According to the author, flow is a mental state that when the user is executing an activity in which he/she is immersed, feeling focused, completely involved and fulfilled during task execution. So,

this model takes into account the psychological steps that players reach during gameplay. This occurs in an way that the main goal is controlling the challenge levels aiming to maintain the player inside the flow, avoiding to reach boredom (no challenges at all) or frustration (challenges are too hard). Figure 1 show a graph of flow theory presented by Csikszentmihalyi [17].



The Flow. After Mihaly Csikszentmihalyi, *The Flow* (1990), p. 74

Figure 1. Demonstration of flow theory, by Csikszentmihalyi.

The model presented by Csikszentmihalyi shows how a task difficulty is directly related to the perception of who is executing it. The flow channel illustrates that difficulty can be progressively improved, since there exists time to the player to learn and improve his/her skills to overcome this challenge. Thereby, this model avoids frustration of very hard situations or boredom caused by very easy situations. Furthermore, Csikszentmihalyi and Nakamura [19] goes beyond and determine that the ratio of challenges to skills should be around 50/50 in order to produce enjoyable experiences.

On the other way, there are some studies that question how valid the ratio of challenges to skills really is as a measure of flow. Løvoll and Vittersø [20], for instance, presents a work with some empirical evidence that contest the idea that flow is produced when challenges and skills are harmonized. According to them, the interaction between challenges and skills as independent variables gave no support to the challenge skill ratio proposed by Csikszentmihalyi and Nakamura.

In a different approach, if we can balance the fantasy, control, challenge and curiosity proposed by Malone [16] and associate it to the progressive development of difficulty presented in flow model by Csikszentmihalyi [17], it is possible that the resulting game can entertain the player. However, using just these features does not show if game challenges are compatible with player skills. So, it is necessary measuring techniques to define when and how difficulty should be adjusted.

#### A. Evaluating the Difficulty Level

According to Andrade et al. [11], there are some different approaches to dynamically balance the difficulty level of a game. However, all of these approaches require measuring,

implicitly or explicitly, the difficulty level that the player is facing on that moment. Defining player difficulty level is crucial to game mechanics evaluation and possible adjustments. This measurement can be done by using heuristics, for example the success rate of skill landing, the capture of enemy points, the time used to complete a task or any other metric that can evaluate the player. Missura and Gärtner [21] made a relation between game runtime, health and score in a way that it composes an evaluation criteria that performs the game difficulty adjustment. Demasi and Adriano [22] developed a heuristic function called “Challenge Function” that is responsible for describing the game state, and tries to show how hard the game is for the player in a given time.

Another way to track difficulty levels is using body language. Van Den Hoogen et al. [23] mentions that body language of a player could be related to his/her experience during play. According to the authors, there are evidences that show that specific postures, facial expressions, eye movements, stress over mouse/keyboard/joystick, and others, could evidence experiences like interest, excitement, frustration and boredom. For player experience evaluation, authors provide a monitoring ambient, in this place there where pressure sensors in the chairs and mouse. Also cameras were placed to register movements and facial expression. The results of this experiment show that the behaviors observed are directly related to the excitement level and dominance felt during the game. Nacke e Lindley [24], besides using cameras to capture body language, also used electrodes to track mental reaction from players during a First Person Shooter (FPS) match. The results obtained during player monitoring were based in flow theory proposed by Csikszentmihalyi [17], therefore, authors could observe if the players were inside the flow, anxious or bored during the gameplay.

Although the explicit measuring (external monitoring) of difficulty levels could provide fine results related to game adaptations to player’s skill, it is impracticable to the dynamic difficulty adjustment. This can be observed because not all players have measuring tools at home and using such tools could be intrusive, since this could make the player uncomfortable by being monitored. Implicit approaches (metrics and heuristics) do not need external equipment and therefore these approaches are more popular among game developers. Besides, implicit approaches favors the conditions that players must not perceive that difficulty is being adjusted during gameplay.

This paper tries to perform a dynamic difficulty adjustment through the development of a mechanism that switches between three distinct artificial intelligences in order to provide an opponent that better suits the player’s abilities. The mechanism perform several evaluations during the match indicating moments where the game is unbalanced and then execute the difficulty adjustment.

### III. DEFENSE OF THE ANCIENTS

The game DEFENSE OF THE ANCIENTS (DotA) is a Multiplayer Online Battle Arena (MOBA) version of the game WARCRAFT III: REIGN OF CHAOS and later to its expansion, WARCRAFT III: THE FROZEN THRONE. The scenario objective is for each team to destroy the opponents’ Ancient, a heavily guarded structure at opposing corners of the map. Players use powerful units known as heroes, and are assisted by allied heroes (played by others users) and AI-controlled fighters known as creeps. As in role-playing games, players level up their heroes and use gold to buy items and equipment during the mission.

According to Johnson et al. [25], MOBA games were found to offer less autonomy, more frustration and more challenges. These findings with respect to autonomy seems most likely to be a function of the fact that MOBA games involve fairly focused competition with other players. Moreover, the greater levels of frustration experienced may also be a function of the focused competition that occurs in MOBA games and the steep learning curve. With less focus on the immersive qualities of the game and greater focus on competing and cooperating with others, there is more potential for frustration with the performance of others players. This interpretation is supported by players reporting a greater challenge when playing MOBA games. Due to these characteristics, the use of a mechanism that performs the difficulty balance dynamically seems to be a viable alternative to minimize and/or avoid that such frustrations be experienced by the players. Therefore, the game DEFENSE OF THE ANCIENTS (DotA) was chosen to be the testbed of this work.

#### A. Gameplay

To provide challenges that suit the player’s skills it is necessary to comprehend the gameplay that involves the game. The DotA game can be summarized into two teams playing against each other: the Sentinel and the Scourge. Players on the Sentinel team are based at the southwest corner of the map, and those on the Scourge team are based at the northeast corner. Each base is defended by towers and waves of units (creeps) which guard the main paths leading to their base. In the center of each base is the “Ancient”, a building that must be destroyed to win the game.

Each player controls one hero, a powerful unit with unique abilities. In DotA, players on each side choose one of 110 heroes, each with different abilities and tactical advantages over other heroes. The scenario is highly team-oriented; it is difficult for one player to carry the team to victory alone. The DEFENSE OF THE ANCIENTS game allows up to ten players in a five-versus-five format.

Since the gameplay goes around strengthening individual heroes, it does not require focus on resource management and base-building, unlike most traditional real-time strategy games. When killing enemy units or neutral units, the player

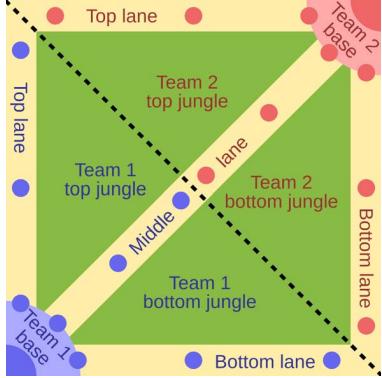


Figure 2. General map from MOBA games.

gains experience points and when enough experience is accumulated the player increases his/her level. Leveling up improves the hero's toughness and the damage it inflicts, and allows players to upgrade spells or skills. In addition to accumulating experience, players also manage a single resource of gold.

In DotA, besides the small periodic income, heroes can earn gold by killing hostile units, base structures, and enemy heroes. With gold, players can buy items to strengthen their hero and gain abilities. Also, certain items can be combined with recipes to create more powerful items. Buying items that suit one's hero is an important tactical element of the game.

The DotA game also offers a variety of game modes, selected by the game host at the beginning of the match. The game modes dictate the difficulty of the scenario, as well as whether people can choose their hero or are assigned one randomly. Many game modes can be combined, allowing more flexible options.

### B. Map

The map is segmented into three different lanes, the top, the bottom, and the middle lane. Each one of these lanes leads to the other team's base, guarded by towers along the way. During the early laning phase of the game, most gameplay is centered around the lanes. Figure 2 represents a general MOBA map with its lanes, bases and towers along each lane.

The map area located between the lanes is called jungle. This is where neutral creeps can be found, which can be killed for gathering more gold and experience points. It is possible to level up by killing creeps in the jungle instead of in the lanes. This practice is called jungling.

Each team has defensive towers placed along the lanes leading to the Ancient. Towers inflict single target damage to heroes and creeps. In the early stages of the game, a hero can only take a few hits from a tower before dying, so one must be careful as to not get too close to towers until they

have gained enough strength. In the Figure 2 the towers are represented by little circles placed in the lanes.

### C. Game Adaptations

To use the game DEFENSE OF THE ANCIENTS as a testbed, some adaptations were made in order to better suit the needs of this work. As mentioned before, the original game allows the player to choose his/her hero among 110 different options. But, for this work, we chose to restrict this quantity to only 10 heroes, equally distributed between both teams.

Each hero has distinct characteristics, behaviors and abilities. Thereby, to better focus on the strategies and the development of abilities, we designed our artificial intelligence to control one specific hero. The selection performed were random and the chosen character is *Lion - The Demon Witch*. Given this choice, it became possible to classify which abilities and behaviors should be implemented so that the artificial intelligence would work with a consistent behavior during the game match.

## IV. METHODOLOGY

Our difficulty adjustment mechanism consists in the development of three different types of artificial intelligence that will be chosen during the match in order to present challenges that suit the player's skills. To select the right opponent, a difficulty evaluation is perform during the game and if it indicates that the players are not evolving in the same pace, it executes the necessary adjustment. Throughout this section, we shall address the artificial intelligence developed, the game features, the difficulty evaluation process and the mechanism to dynamically adjust the presented difficulty during a match.

### A. Artificial Intelligence

To be able to provide an opponent that can face different skilled players, the artificial intelligence must be implemented with distinct ability levels to simulate the most different behaviors played. Since the artificial intelligence must simulate an opponent player, the developed algorithm implements actions and behaviors to a hero unit. During a game match this hero should follow the player's performance, so if the player is having a good evolution, the hero controlled by artificial intelligence must be able to also do the same. However, if the player is not evolving enough or if his/her development started to decrease, the artificial intelligence hero must also decrease its strategies and skills and keep up with its opponent.

The hero behavior was divided into three categories: easy mode, regular mode and hard mode. Each one of these categories has singular aspects that aims to be suitable to players with different abilities. These modes will be described as follows.

*Easy Mode::* In the easy mode, the hero performs regular attacks every time an enemy enters in its attack range. When an allied tower is under attack, the hero detect the need for defense and moves towards the attacked ally in order to defend it. Another strategic action is how the hero chooses the enemy tower to be its main target. Every time the hero starts a moving action, it analyses which one of the enemy's tower has taken more damage and is closer to be defeated. Once it finds, the hero sets that tower as the main target and go in that direction. It is important to mention that, in the easy mode, all the attacking actions that the hero performs are basic attacks. The hero also retreats as a defense strategy. So when its health points are below 30%, it starts to retreat towards the allies base, where it can recover its health when it reaches a specific recovery building. The easy mode was created for beginners or some less skilled players, where the implemented strategies are not very complex and does not use any special character skill (also known as spells).

*Regular Mode::* In the regular mode, besides the strategies implemented for the easy mode, the hero also starts to manipulate items. The item manipulation is very helpful to improve the hero's attributes and also to recover some attributes that has been decreased, for example, items to recover health points or mana. Likewise, there are items to increase attributes like strength, speed, intelligence, among others. As part of the defense strategy, if the hero's health points reach 30% or less, it will first use some health potion to recover it and if these items are over, then the hero starts to retreat towards its base. The regular mode was created to cover those players that have already some experience and know how to use some of the game functionalities in his/her favor but are not experts yet.

*Hard Mode::* The hard mode has all the strategies implemented on both preceding modes, besides its own specific actions. Here, the hero goes beyond item manipulation and starts to learn, improve and cast spells. Spells are unique skills that each hero has. These spells can give a more effective damage on the enemy, can boost the recovery of its own attributes (like mana or health points), can give some kind of advantage to allied units (like freezing the enemies), among other possibilities. Every time the hero gains a new level it also gains one attribute point to distribute among its spells. So in this mode, besides the regular attack, the hero also casts spells to attack enemies or defend allies. Here we also decide to implement a new strategy for a head-to-head combat. In order to avoid losing the combat against another hero, the artificial intelligence algorithm keeps monitoring the area around its hero. Therefore, if an enemy hero enters the monitored area, the hero controlled by the artificial intelligence will take advantage on that and will begin to attack it. The strategies to defense allied towers and to retreat are the same developed on regular mode. The hard mode was created to cover those players that have more experience

|                  |                      | Artificial Intelligence |              |           |
|------------------|----------------------|-------------------------|--------------|-----------|
|                  |                      | Easy Mode               | Regular Mode | Hard Mode |
| Defense Strategy | Defend Allied Towers | X                       | X            | X         |
|                  | Retreat              | X                       | X            | X         |
|                  | Item Manipulation    |                         | X            | X         |
|                  | Main Attack          | X                       | X            | X         |
|                  | Target Selection     | X                       | X            | X         |
|                  | Track Enemy Hero     |                         |              | X         |
|                  | Cast Spell           |                         |              | X         |

Figure 3. Summary of the developed strategies for each artificial intelligence.

on the DotA game and also know how to use the game functionalities in their favor. This kind of player may be an expert on the game or a quick learner.

The table displayed in Figure 3 summarizes all the developed difficulty modes and their strategies.

#### B. Difficulty Evaluation Process

A difficulty evaluation process was elaborated to be performed during the game and indicate when the players are not evolving in the same pace. For that, it was necessary to observe which game features should be analyzed and how to properly use the information from each one of them. The analyzed features and the evaluation process will be described below.

*1) Game Features:* To evaluate a game match it is crucial to identify which features can represent the players' performance and should be considered relevant to the evaluation. In our testbed, we identified three important features that can illustrate the player's behavior during a DotA match. These features are: Hero's Level, Hero's Death and Towers Destroyed. Each one of these features will be described below:

*Hero's Level::* This feature represents the player's evolution during a match, where the greater is the level value, the stronger is the character. Although this feature represents the evolution, it should not be the only analyzed feature because it is possible that the player increases his/her hero's level without really increasing his/her abilities. For example, the player can keep the hero closer to battles without engaging in any fight and, by doing that, it will gain some experience points that are shared among the allies that are closer to the battle and it will help the hero to evolve its level. Thereby, even if all players have heroes with equivalent levels, this feature alone does not give a real track on the game balance.

*Hero's Death::* This feature is responsible for showing how many times the hero has died during a DotA match. Differently from all other features, the hero's death may represent the player's performance and the level of difficulty that he/she is facing more accurately. For example, an inexperienced player, even having a hero with a high level, may have a high death rate, since he/she may not know how

to use more properly the characteristics and peculiarities of his/her character as well as a possible lack of game strategies. Thereby, this feature seems to represent more accurately how well the player is facing the game challenges.

*Towers Destroyed::* This feature is the amount of enemy's towers destroyed by the allied team. It represents the team expansion and dominance over the map. Although this feature is not directly related to the player's performance, since other allies can also destroy towers, it gives us a good notion of the game's progress and team expansion over the map. Therefore, if a team is quickly progressing over the map, it may represent that the game is unbalanced.

2) *Difficulty Evaluation Process:* In order to perform a dynamic difficulty adjustment, it is necessary to evaluate the game from time to time and verify if the game is presenting challenges suitable to the player's performance. If the player is having a poor performance, the game should be capable to identify that and reduce its difficulty. In the same way, if the player evolves faster than the challenges presented, the game should increase its difficulty.

Once we have defined the game features that must be analyzed, this process can be summarized into a creation of an heuristic function that will keep track on the player's performance and inform when it is necessary to adjust the difficulty. This heuristic function will be our evaluation method during the game match and from now on it will be called as evaluation function. So, considering the features mentioned before and the impact that each one represents on the player's performance, we have:

$$P(x_t) = H_l - H_d + T_d, \quad (1)$$

where  $P(x_t)$  is the performance function of player  $x$  on time  $t$ .  $H_l$  is the hero's level,  $H_d$  is the hero's death and  $T_d$  is the towers destroyed. It is important to mention that the values of this features are related to the player and his/her hero. Once you have computed the performance value of the player from start time ( $t = 0$ ) to current time ( $t = i$ ), it is necessary to do the same for time  $t = i - 1$ . After having both values, it is possible to calculate the current evolution of the player, as shown in the equation below:

$$P'(x) = P(x_t) - P(x_{t-1}). \quad (2)$$

Once the performance function was calculated for both players ( $x$  and  $y$ ) the evaluation value can be obtained by:

$$\alpha = P'(x) - P'(y), \quad (3)$$

where  $\alpha$  is the difference among performances. It is important to mention that player  $x$  is the one that we are analyzing and player  $y$  is the one controlled by the artificial intelligence system. Therefore, the player  $y$  is the one that will have its difficulty adjusted during the game.

### C. Dynamic Difficulty Adjustment Mechanism

The proposed mechanism is the key to make the adjustment work properly during the game. Until now we have only showed how to verify if the player's performance is balanced to a certain opponent or not. Thereby, the main task

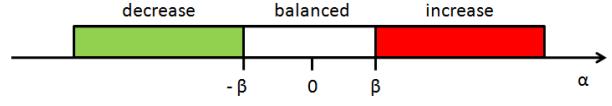


Figure 4. The verification performed by the adjustment mechanism.

of the implemented mechanism is to analyze the  $\alpha$  value and perform or not the difficulty adjustment at the game time  $t$ .

The mechanism works by evaluating the  $\alpha$  variable and constantly verifying if this variable is within the  $\beta$ 's range. Where  $\beta$  represents the limit value of the evaluation function. This value means how far a player can perform better than the other player, without considering the game unbalanced. If the value of  $|\beta|$  is a huge number, then the adjustment will occur with less frequency, since it may take some time to  $\alpha$  overcome  $\beta$ . Likewise, if  $|\beta|$  is a small number, then the adjustment will occur more frequently, since it may overcome  $\beta$  more easily. And if  $\alpha$  stays inside the limits values of  $-\beta$  and  $\beta$ , it means that both players are having a similar performance and therefore, the match is currently balanced. The Figure 4 illustrates this approach.

## V. EXPERIMENTS

In order to verify the effectiveness of the proposed mechanism, a series of experiments was performed. The players' performance were analyzed along with the behavior of their heroes. The dynamic adjustment mechanism was also observed, as well as its variations and the impact caused on the matches.

On each experiment we ran the game with the static artificial intelligence controlling one team against the dynamic artificial intelligence controlling the other one. Was performed a game set with 20 matches for each experiment and after observing the results contained in the gamelog of several matches we defined that the  $\beta$  limits should be  $-1$  and  $1$ . Therefore, every time the difference among performances ( $\alpha$ ) exceeds the  $\beta$  limits, the difficulty of the dynamic AI should be modified accordingly to the obtained value.

### A. Baseline

First, we performed an unbalanced match in order to stipulate a baseline to compare with the obtained results from all three experiments. This baseline match is set by two different artificial intelligence players with static behavior. One of them is on easy mode, representing a player without experience, and the second one is on hard mode, representing a very experienced player. The results of the mentioned match are shown in Figure 5, where the difference among both performances can be noticed.

The player's performance is measured taking into account his/her current state during the match. The positive peaks

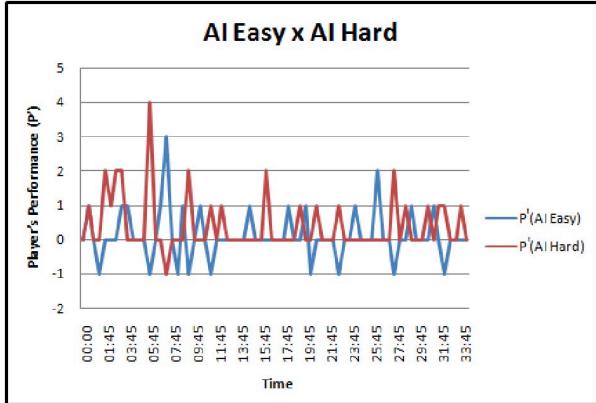


Figure 5. Graphic with baseline values obtained from a match with static difficulty

represents moments where the player improved his/her abilities when compared to his/her last state. Likewise, negative peaks means that the player had his/her performance decreased based on his/her last game state. Converting these to game situations, when a hero gains a level or the team manage to destroy a tower, then this will impact positively in his/her development, increasing the player's current performance. Similarly, if a hero dies this will result in a negative impact in his/her development decreasing the player's current performance.

During this game match, the hard mode player kept increasing his performance, presenting only one time of regression in his development. Meanwhile, the easy mode player performance was very unstable, presenting many moments of regression in his development. Therefore, we can consider that a match will be balanced if the difference among both performances were not divergent. So, examining once again the graphic, it is possible to observe that each performance peak shows itself as an appropriate moment to execute a difficulty balance in order to get the players' performance closer to each other.

Figure 6 shows the accumulative performance value from each player during this particular match. On this graph, it becomes clear that the hard mode player evolves much faster than the easy mode player. This greater performance evolution can be related to the fact that the hero increases his level rapidly and has a low amount of fatalities. Differently from the easy player that although his hero had a great level development, the amount of deaths was also high, leading to a poor performance when compared to the hard mode player. Therefore, due to that difference between them, the adjustment appears to be necessary in order to minimize this disparity among their behaviors and present a more fair and competitive game.

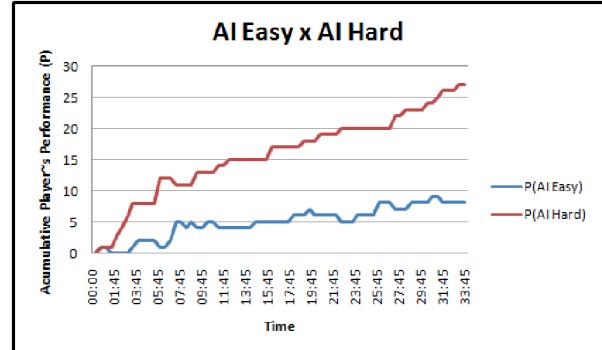


Figure 6. Graphic with the accumulative player's performance ( $P$ ) value.

### B. Easy x Adaptative

The experiments were performed using the artificial intelligence developed to control the heroes, one from each team. For player A we decided to use a static artificial intelligence in order to simulate the behavior of a human player. For player B we applied the proposed mechanism, where this player should keep its abilities suitable to player A and for that it should perform a dynamic difficulty adjustment. The first set of experiments, we manage to simulate a beginner player with player A. The player B started on regular mode and during the match it should be balanced to better fit the skills of player A. Figure 7 shows the performance of both players during this match ( $P'$ ), while Figure 8 shows the results of the evaluation function ( $\alpha$ ) and the difficulty adjustments made during the game.

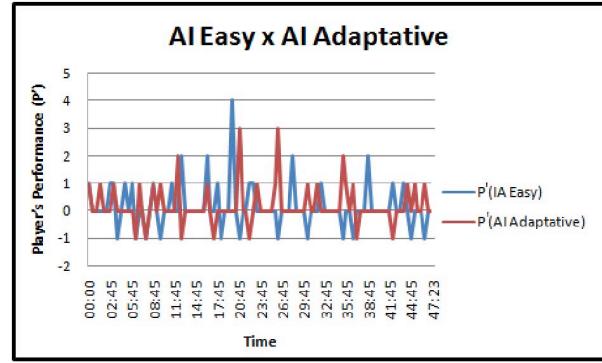


Figure 7. Graphic with players performances during one match.

As mentioned before, the player's performance is measured by taking into account his/her current state during the match. The positive peaks represents moments where the player had improved and negative peaks means that the player had decreased based on his/her last game state. Figure 7 shows that the adaptative artificial intelligence (player B) managed to keep its performance similar to its opponent, the easy mode player A.

On Figure 8 we can track how well the adaptative player

(player B) manage to be compatible with player A during the match. When the evaluation function shows negative peaks, it means that the difficulty should be adjusted and decreased by one level. Likewise, if there are positive peaks resulted by the evaluation process, than the difficulty of the adaptative player should be increased by one level. Moments where the evaluation function remains constant (equals 0) means that the performance of both players are very similar and due to that no adjustment is necessary at this time. Therefore, the difficulty can be maintained.

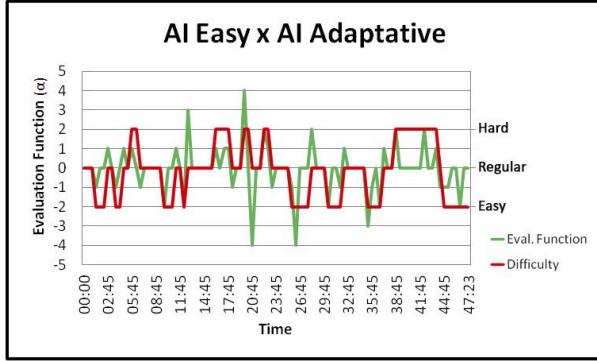


Figure 8. Graphic with the difficulty adjustments performed by the mechanism during one match.

It is important to mention that the difficulty adjustment is performed by increasing or decreasing one level of each time. With this approach we minimize the possibility of the opponent player noticing the behavior change. After analyzing this set of experiments and study the gamelogs obtained from each one, we observed that in 85% of the matches, the adaptative player B managed to keep the game balanced and as result of each match, player A won 60% of the matches and player B won 40%.

#### C. Regular x Adaptative

On the second set of experiments, we kept using the artificial intelligence developed to control two players, one from each team. Here, we manage to simulate an intermediary player with player A using a static artificial intelligence on regular mode. For player B we applied the proposed mechanism, starting it on regular mode and during the match it should keep the game balanced. Figure 9 shows the performance of both players ( $P'$ ) during one single match. Likewise, Figure 10 shows the results of the evaluation function ( $\alpha$ ) during the game and the difficulty adjustments made over the match.

The analysis performed in this set of experiments is pretty similar to the previous one. The positive peaks represents moments where the player had improved and negative peaks means that the player had decreased its performance. In Figure 9 we can observe that the adaptative artificial intelligence (player B) tried to follow its opponent's performance (player

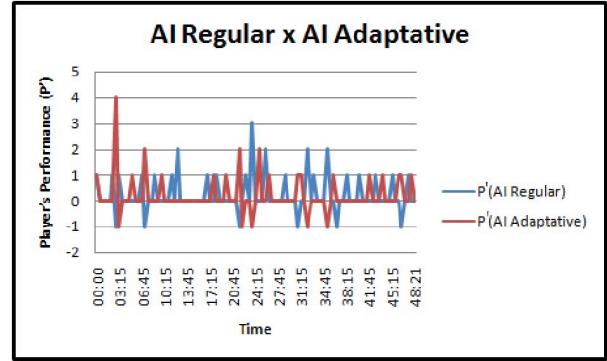


Figure 9. Graphic with players performances during one match.

A) presenting similar peaks at the same time, or in some moment closer, to its opponent.

On Figure 10 we can follow all the adjustments made during the match. The adaptative player spent most of its time alternating between the regular mode and the hard mode. This variation can be understood as moments where the player B were having a poor development when compared to player A, and the need of increasing the difficulty was perceived. Similarly, when player's B behavior were standing out the need for reducing the difficulty could also be seen. The graphic also shows that player B stayed balanced during the game. Furthermore, after analyzing this second set of experiments and study all gamelogs collected, we observed that the players had a compatible performance in 90% of the matches. The results of the matches can be splitted into a 50-50 victories.

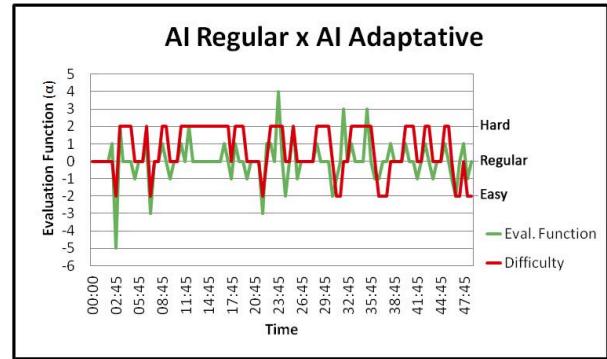


Figure 10. Graphic with the difficulty adjustments performed by the mechanism during one match.

#### D. Hard x Adaptative

On the last set of experiments, we manage to simulate an expert player (player A) against the developed adaptative player (player B). As we mentioned before, the adaptative player started on regular mode and changed its behavior during the match in order to keep the game balanced.

Figure 11 shows the performance ( $P'$ ) of both players during one match. Likewise, Figure 12 shows the results of the evaluation function ( $\alpha$ ) during the game and the difficulty adjustments made over the match.

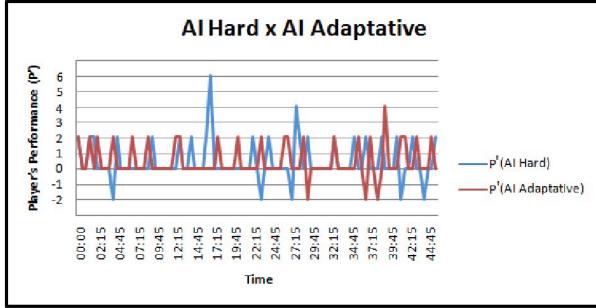


Figure 11. Graphic with players performances during one match.

Analyzing the results from Figure 11, the adaptative player started developing a better performance than player A in the beginning of the match. Therefore, it was detected that the difficulty should be reduced in order to keep the balance (Figure 12). After that, they keep their performances very close and the difficulty keeps alternating between easy mode and regular mode until player A can present himself/herself better/stronger than player B. The opposite can also be seen, when player B keeps alternating between regular mode and hard mode in order to reach player's A performance.

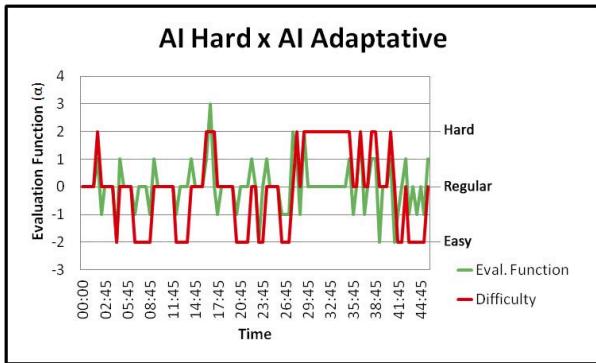


Figure 12. Graphic with the difficulty adjustments performed by the mechanism during one match.

Furthermore, after analyzing the gamelogs collected from each game, we observed that the adaptative player (player B) changed its difficulty and succeed to keep the match balanced on 80% of the experiments. As result of the battles, player A won 45% of the matches.

#### E. Discussion

Not all the cases presented the expected results, which has resulted in unbalanced matches. To get to this conclusion, we observed all the executed matches and studied all the collected gamelogs. These gamelogs kept track of the game

on every 15 seconds, recording the current situation of both teams, the related features, their values, among other information. Once the game was finished, we started to translate those collected information, comparing the values from both heroes and making the necessary assumptions.

Considering all the performed experiments 10% of it were unbalanced because the mechanism took too long to perform each adjustment, leading to a great difference between the players performance. So, when the players were getting closer to a balance, the match has ended. On the other way, 5% of the executed experiments were unbalanced due to an excess of adjustments. In these scenarios, the adjustments were being performed too quickly, leading player B to not evolve properly during the match, which resulted an easy game for player A.

| AI Adaptative |     |      |
|---------------|-----|------|
|               | Win | Lose |
| AI Easy       | 8   | 12   |
| AI Regular    | 10  | 10   |
| AI Hard       | 11  | 9    |

Figure 13. Final results from all matches performed on the experiments.

After performing all the experiments it was possible to summarize the obtained final results from the game matches. Figure 13 shows the amount of victories and losses of the adaptative AI against the easy, regular and hard modes. According to these values, we can observe that the game kept impartial once both players had very close results, showing that exist the possibility of the human player win or lose the match, it will rely on his/her abilities.

#### VI. CONCLUSION

The dynamic difficulty adjustment consists in an alternative towards the definition of the game challenge levels. This adjustment is dynamically performed, making it possible to track the player's skills and adjust itself during game runtime.

The presented work aimed to increase the player's entertainment by providing a mechanism that adjusts the game AI according to the player's skills. This mechanism was implemented on a game modification of WARCRAFT III, called DEFENSE OF THE ANCIENT (DotA). After performing experiments that simulate the three main player's behaviors (beginner, regular and experienced), it was possible to verify that the dynamic difficulty adjustment mechanism was able to keep up with the player's abilities on 85% of all experiments. On the remaining experiments that failed to suit the player's skill, 10% of it occurred because the adjustment mechanism spent too much time to perform each needed adjustment which leaded to a great difference between the players performance. And the last 5% of it occurred due to

an excess of adjustments that was performed too quickly, without giving enough time to the game to evolve properly.

Given the presented results, we can conclude that the proposed mechanism behaved as expected and is capable to offer a game match compatible with the simulated player's performance. Also, after observing all obtained results, we can state that the key to a balanced game is to keep changing the difficulty of the adaptative player in order to follow the performance of the human player and avoid boredom and frustration.

As future work, the dynamic difficulty adjustment mechanism will be improved in order to decrease the amount of cases where the balance did not worked properly. We also intend to perform some qualitative experiments on human players with different experiences on the game DEFENSE OF THE ANCIENTS (DotA) to better evaluate the developed mechanism.

## REFERENCES

- [1] P. Thompson, R. Parker, and S. Cox, "Interrogating creative theory and creative work: Inside the games studio," *Sociology*, p. 0038038514565836, 2015.
- [2] A. Nareyek, "AI in computer games," *Queue*, vol. 1, no. 10, p. 58, 2004.
- [3] M. C. Machado, E. P. Fantini, and L. Chaimowicz, "Player modeling: Towards a common taxonomy," in *Computer Games (CGAMES), 2011 16th International Conference on*. IEEE, 2011, pp. 50–57.
- [4] A. M. Smith, C. Lewis, K. Hullett, G. Smith, and A. Sullivan, "An inclusive taxonomy of player modeling," *University of California, Santa Cruz, Tech. Rep. UCSC-SOE-11-13*, 2011.
- [5] D. A. Bowman and R. P. McMahan, "Virtual reality: how much immersion is enough?" *Computer*, vol. 40, no. 7, pp. 36–43, 2007.
- [6] G. N. Yannakakis and J. Hallam, "Towards optimizing entertainment in computer games," *APPLIED ARTIFICIAL INTELLIGENCE*, 2007.
- [7] B. B. P. L. de Araujo and B. Feijó, "Evaluating dynamic difficulty adaptivity in shoot'em up games," in *Proceedings of the XII Brazilian Symposium on Games and Digital Entertainment - SBGames 2013*, São Paulo, Brazil, Oct. 2013, pp. 229 – 238.
- [8] R. Koster, *Theory of fun for game design*. O'Reilly Media, Inc., 2010.
- [9] M. Mateas, "Interactive drama, art and artificial intelligence," 2002.
- [10] R. Hunnicke, "The case for dynamic difficulty adjustment in games," in *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology*. ACM, 2005, pp. 429–433.
- [11] G. Andrade, G. Ramalho, H. Santana, and V. Corruble, "Extending reinforcement learning to provide dynamic game balancing," in *Proceedings of the Workshop on Reasoning, Representation, and Learning in Computer Games, 19th International Joint Conference on Artificial Intelligence (IJCAI)*, 2005, pp. 7–12.
- [12] R. A. Bartle, *Designing virtual worlds*. New Riders, 2004.
- [13] G. Xavier, "A condição eletrolúdica: Cultura visual nos jogos eletrônicos," *Teresópolis: Novas ideias*, 2010.
- [14] C. Klimmt, C. Blake, D. Hefner, P. Vorderer, and C. Roth, "Player performance, satisfaction, and video game enjoyment," in *Entertainment Computing-ICEC 2009*. Springer, 2009, pp. 1–12.
- [15] G. N. Yannakakis, "How to model and augment player satisfaction: a review," in *Proceedings of the 1st Workshop on Child, Computer and Interaction (ICMI'08)*, ACM Press, Montreal, Canada, 2008.
- [16] T. W. Malone, "Toward a theory of intrinsically motivating instruction\*," *Cognitive science*, vol. 5, no. 4, pp. 333–369, 1981.
- [17] M. Csikszentmihalyi, *Flow*. HarperCollins, 1991.
- [18] S. Egenfeldt-Nielsen, J. H. Smith, and S. P. Tosca, *Understanding video games: The essential introduction*. Routledge, 2013.
- [19] M. Csikszentmihalyi and J. Nakamura, "Effortless attention in everyday life: A systematic phenomenology," 2010.
- [20] H. S. Løvoll and J. Vittersø, "Can balance be boring? a critique of the challenges should match skills hypotheses in flow theory," *Social indicators research*, vol. 115, no. 1, pp. 117–136, 2014.
- [21] O. Missura and T. Gärtner, "Player modeling for intelligent difficulty adjustment," in *Discovery Science*. Springer, 2009, pp. 197–211.
- [22] P. Demasi and J. d. O. Adriano, "On-line coevolution for action games," *International Journal of Intelligent Games & Simulation*, vol. 2, no. 2, 2003.
- [23] W. Van Den Hoogen, W. Ijsselsteijn, and Y. de Kort, "Exploring behavioral expressions of player experience in digital games," in *Proceedings of the workshop on Facial and Bodily Expression for Control and Adaptation of Games ECAG 2008*, 2008, pp. 11–19.
- [24] L. Nacke and C. A. Lindley, "Flow and immersion in first-person shooters: measuring the player's gameplay experience," in *Proceedings of the 2008 Conference on Future Play: Research, Play, Share*. ACM, 2008, pp. 81–88.
- [25] D. Johnson, L. E. Nacke, and P. Wyeth, "All about that base: differing player experiences in video game genres and the unique case of moba games," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, 2015, pp. 2265–2274.

## Article

# Mechanics and Metagame: Exploring Binary Expertise in League of Legends

Games and Culture  
2017, Vol. 12(5) 426-444  
© The Author(s) 2015  
Reprints and permission:  
[sagepub.com/journalsPermissions.nav](http://sagepub.com/journalsPermissions.nav)  
DOI: 10.1177/1555412015590063  
[journals.sagepub.com/home/gac](http://journals.sagepub.com/home/gac)



Scott Donaldson<sup>1</sup>

## Abstract

This article examines the significance of two types of expertise in the popular multiplayer video game *League of Legends*. Previous research into multiplayer games has explored a variety of expertise models, some which concern only a player's mastery of the controls and some which take negotiation of a game's sociocultural context into account. This article analyzes play in *League of Legends* through the lens of a binary model of expertise, outlining examples of the in-game and out-of-game practices used by players in their pursuit of competitive success. I argue that forms of out-of-game or "metagame" expertise are of particular importance in *League of Legends* and are of such depth that further research would be highly valuable.

## Keywords

*League of Legends*, expertise, metagame, game mechanics, multiplayer

## Introduction

As demonstrated by a number of studies on video gaming, play does not take place in a sociocultural vacuum, as players will always draw upon existing knowledge derived from previous play experiences or engagement with extrinsic resources in

---

<sup>1</sup> School of Media, Culture and Creative Arts, Curtin University, Perth, Australia

### Corresponding Author:

Scott Donaldson, School of Media, Culture and Creative Arts, Curtin University, Kent Street, Perth 6102, Australia.

Email: scott.donaldson1@postgrad.curtin.edu.au

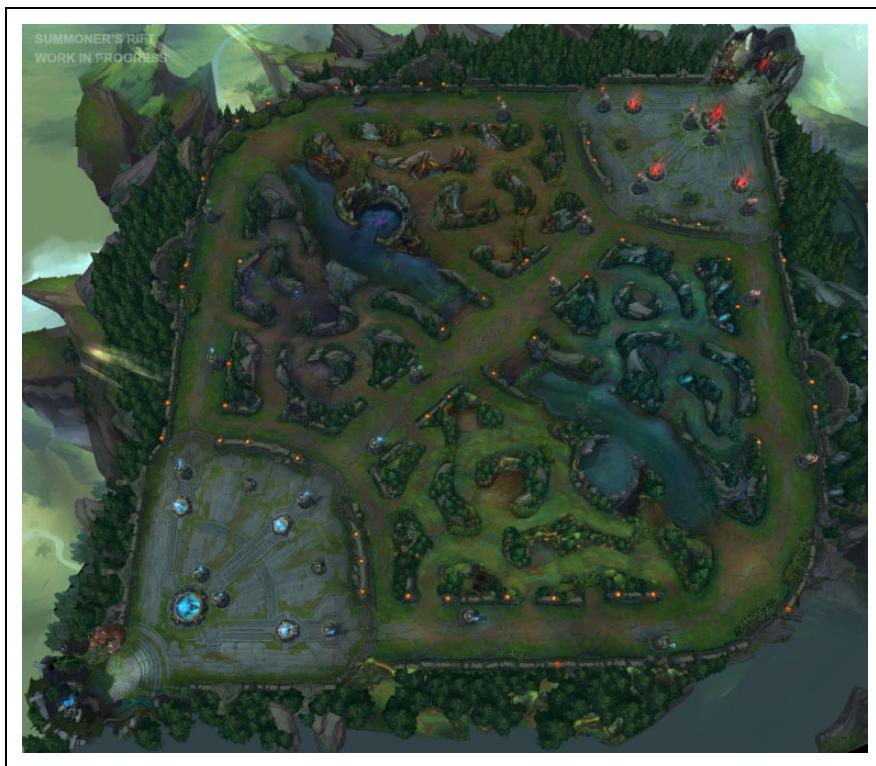
order to achieve success (Ang, Wildon, & Zaphiris, 2010; Chen, 2010; Harper, 2010; Humphreys & de Zwart, 2012; Jansz & Martens, 2005; Lynch, 2013). Chen's (2010, p. 52) ethnographic study of popular massively multiplayer online role-playing game *World of Warcraft* explores this idea, identifying that a player will likely not find success in the latter stages of the game unless he or she acquires some level of social expertise. He separates the game into two parts, divided by the maximum character level, with the acquisition of what I will refer to as "mechanical expertise"—knowledge and mastery of the game mechanics—forming a substantial portion of the "forgiving early game content" and the acquisition of social capital as a barrier of entry to the "technically difficult" post-60 content, which requires large-scale cooperation and coordination between players. A binary division of expertise is also alluded to by a number of other theorists, such as Jakobsson, Pargman, and Rambusch (2007, p. 158) in their case study of gameplay in the first-person shooter game *Counter-Strike*. They refer to the two categories of gameplay as the player's "handling of the game," which concerns the "physical and motorical" activities of gameplay, and the player's "meaning-making activities," which involves his or her "understanding of the game in terms of how the game is to be played, their role in the game and the culture around the game." Neuenschwander (2008, p. 190) notes a similar concept in the "multifaceted proposition" of learning a tabletop role-playing game, with the first stage being the accumulation of knowledge of the game rules as they are described in the rulebook and the second stage being the understanding of social rules, which might both deviate from the rulebook and differ from playgroup to playgroup. A common factor among these games is that they are either exclusively multiplayer or intended for multiple players, and that the second form of expertise concerns their respective sociocultural contexts. Single player games typically do not require this form of expertise. In their analysis of "learning curves" of single player games, Przybylski, Rigby, and Ryan (2010, p. 156) find that "mastery of controls" is the single most important factor in how a player masters such a game. The accumulation of expertise related to that, which is extrinsic to game mechanics, can certainly be referred to as being part of a "metagame," referred to by Salen and Zimmerman (2003, p. 481) as "the relationship between the game and outside elements, including everything from player attitudes and play styles to social reputations and social contexts in which the game is played." We can therefore group the extrinsic forms of expertise discussed by Chen, Neuenschwander, and Jakobsson et al. as forms of metagame expertise, which might be acquired alongside or after the accumulation of the basic level of mechanical expertise (which also differs from genre to genre) necessary for participation.

In addition to its expression as a binary model, expertise in multiplayer games has been theorized as both single- and multi-modal forms. Reeves, Brown, and Laurier offer their own exploration of expertise in *Counter-Strike* (2009, p. 213), although unlike Jakobsson et al. (2007), they find that a player's mechanical abilities are the sole measure of expertise, stating that, among these, "moving competently" is the "crux of playing well." In doing so, they make only passing references to server

selection and in-game weapon selection, decisions which are often based on the “outside elements” described by Salen and Zimmerman (2003). While highly skilled players may be able to cope with high latency brought about by poor server selection (Henderson, 2001, p. 10), a strong negative correlation has been observed between latency and in-game kills in first-person shooter games (Armitage, 2003, p. 140), meaning that correct server selection in *Counter-Strike* will provide a small benefit to the player regardless of their mechanical ability. They also observe that the gameplay of a single match will change over the course of any number of rounds, as players reflect upon rounds played and alter their play style based on their own effectiveness and their ongoing analysis of the enemy team’s play style. This demonstrates the existence of a local, temporary metagame, which, as with server selection, requires from the player a form of expertise related to elements separate from in-game mechanics.

Taylor, Castell, Jenson, and Humphrey (2011) present a multimodal framework for expertise in massively multiplayer online role-playing games (MMORPGs), unique in that it takes into account the various forms of play afforded by the genre. This play-centric (as opposed to player centric) approach means that a player may still be considered an expert even if their focus lies in only a single portion of the game (a player might focus on questing, battling other players, or socializing, for example), a possibility afforded by expansive MMORPGs like *World of Warcraft* and *Guild Wars 2*. The modes of expertise include time and/or resource investment, skill, discourse, and game knowledge (which is subdivided into ludic and narrative knowledge), the acquisition of which can be augmented with third-party game add-ons (McArthur, Peyton, Jenson, Taylor, & de Castell, 2012). While the multimodal, genre-specific framework is certainly effective at describing styles of play, I feel the more general binary approaches of Chen (2010), Neuenschwander (2008), and Jakobsson et al. are more useful as they can both encompass a number of styles of play while remaining applicable to a multitude of multiplayer genres.

In the following, I will use a binary model of expertise to describe competitive play in the multiplayer online battle arena (MOBA) game *League of Legends*, and in doing so will demonstrate the significance of metagame practices in competitive play. Having surpassed *World of Warcraft* as the world’s most popular video game (Riot Games, 2014), *League of Legends* is now receiving a level of academic attention appropriate for its significance in both the eSports industry and contemporary game culture. Existing works have focus on team matchmaking (Kou & Gui, 2014), gender disparity among players (Ratan, Taylor, Hogan, Kennedy, & Williams, 2015), and the relationship between rules and player-created norms (Kou & Nardi, 2014), all of which are significant aspects of not only *League of Legends* and the player culture but the MOBA genre and competitive online gaming in general. This piece will examine and analyze the fundamentals of play in *League of Legends*, thereby providing a solid foundation for further study.



**Figure 1.** Summoner's Rift (2014).

## *League of Legends*

*League of Legends* is a PC-based MOBA, a type of game that combines elements of role-playing, real-time strategy, and tower defense game genres. A single game of *League of Legends* consists of a strategic and oftentimes hectic battle between two teams of five players, each in control of 1 of 124 playable “champions.” A typical game lasts for approximately 30 min, with longer games sometimes surpassing 1 hr.

The game’s primary mode takes place in Summoner’s Rift (Figure 1), a square-bordered zone with a diagonally running river separating the bottom left “Blue” side of the map from the upper right “Red” side. The area in the bottom-left is the blue team’s spawn point (where the team first enters the game) and “nexus”—this is the same for the red team on the opposite side of the map. The two bases are connected by three lanes, known as “Top,” “Mid,” and “Bot.” The mid-lane and river intersect, creating the four wedges of maze-like “jungle,” which make up the rest of the map.

A winning state is brought about by the destruction of the opposing team's nexus, a large structure fortified by defensive towers, which must be destroyed before the nexus can take damage. Five of these towers are located inside the base and there are two in each lane. The towers of only one lane need to be destroyed in order to expose the enemy nexus. While the destruction of the enemy nexus is the main goal, teams will aim to fulfil subgoals, the completion of which is either necessary, such as the destruction of towers, or conducive to victory, such as the slaying of enemy champions. Individual players may too strive to attain personal goals, such as the achievement of a high kill/death ratio or a full inventory. There are also a handful of narrative-based goals that become available when certain conditions are met—for example, if the rival champions Kha'Zix and Rengar are chosen by opposing players, whomever slays their opponent first will be rewarded with a small increase in strength.

The dynamic elements on the map include "minions," "monsters," and the players themselves. Minions are weak, nonplayer characters (NPCs) whose deaths yield gold to the slayer. They spawn in "waves" from both nexuses and march down the lanes, meeting each other in the middle if unimpeded by champions or other obstacles. When in range, minions will mindlessly attack enemy minions, champions, defensive structures, and eventually the nexus itself. Minions are a primary source of income for players, and a substantial portion of the early game consists of "farming"—killing minions so as to earn gold. Monsters are neutral NPCs that populate the jungle and are immobile and harmless unless attacked. As with minions, killing monsters will provide players with a gold income.

Games can generally be divided into three "phases"—laning, mid-game, and late-game. In the laning phase, players will focus on killing minions and monsters in order to gain gold so as to purchase or upgrade equipment. Top lane and mid lane will usually house one laner each, and bot lane will usually house two. Since champions from opposing teams share the lane while farming, skirmishes can often break out between players, although this depends on how willing they are to engage during this phase. It is common that one player will take on the role of "jungler"—this player will not farm minions in lane but rather the monsters in the jungle, emerging into lane in order to kill an enemy laner if they have pushed away from the safety of their own defensive tower. In the mid-game, champions roam the map and group in an attempt to kill large monsters, destroy towers, and pick off enemy champions who stray too far from safety. If one team is "snowballing" (gaining so much gold from killing enemy champions that they become exponentially effective at killing via the items they are able to afford), it is possible for them to win the game during this phase. In the late-game, most champions will be sufficiently geared and will be looking to kill all or most of the enemy team and subsequently destroy their nexus or, if this is not possible, destroy defensive structures or kill Baron Nashor (a powerful monster whose death confers significant stat boosts for the slayer's entire team) if it is available. Once the nexus is destroyed, the game is over and players are placed into the postgame lobby where they can chat and view individual and team statistics.

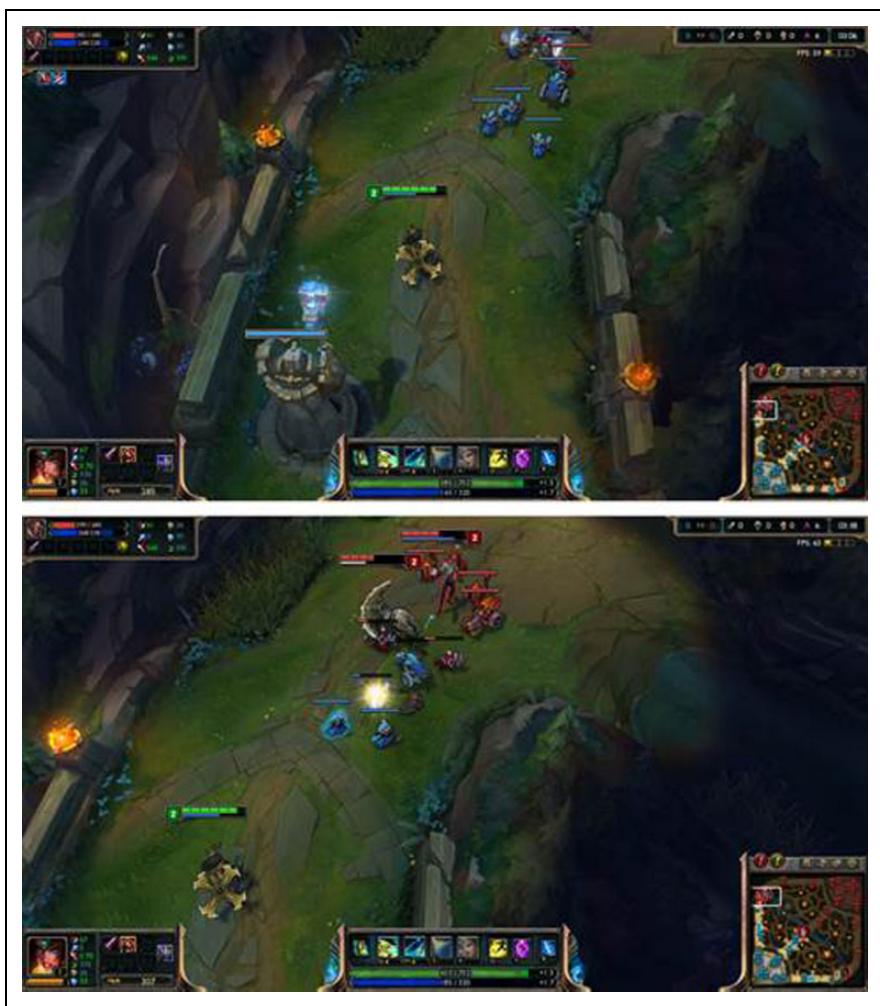
### *League of Legends and Game Mechanics*

A large portion of a player's early time spent with a video game is taken up with learning how the game system works (Taylor, 2012, p. 92). In his ethnographic account of expertise in the MMORPG *World of Warcraft*, Chen (2010, p. 55) describes the initial process of expertise accumulation as one of trial and error, in which new players will experiment with various combinations of items and abilities. Players will also use third-party websites (p. 54) and interface add-ons (p. 56) to hasten their learning and to decrease cognitive load, respectively. Aside from the use of interface add-ons, which are unavailable in *League of Legends*, the process is quite similar since the inbuilt mechanisms for learning are not extensive enough to give players anything beyond a basic understanding of gameplay. New players receive only limited assistance from the game system in developing mechanical expertise—a brief tutorial mode offers players an explanation of the basic control scheme and other basic mechanics, but little attention is given to higher strategy. Each champion also comes with a “recommended” item build, although players must often deviate from this in order to compete effectively against certain enemy team compositions.

That said, players of all skill levels will use external resources and previous experiences to build expertise because no game is played in a vacuum. Experienced gamers new to the MOBA genre will likely recognize the gameplay an amalgamation of role-playing, real-time strategy, and tower defense game genres. Most existing gamers will immediately recognize the health and resource system common in popular role-playing games like *Diablo*, *World of Warcraft*, and *Final Fantasy*, and first-person shooters such as *Halo* and *Evolve*. Role-players will recognize the stat-based method of evaluating in-game strength (a staple of role-playing games since *Dungeons & Dragons*) and the existence of a meta-narrative providing a greater context for the champions and game maps. The control scheme and camera positioning are very similar to that of real-time strategy games, such as *Starcraft II* and *Command & Conquer*, meaning players from these genres will have at least some level of mechanical expertise in this area.

A major example of required mechanical expertise in *League of Legends* is camera control. From a top-down perspective, a player controls his or her champion using a combination of keyboard and mouse actions—although players can reassign controls to suit their preference, players generally move their champion, target abilities, control the camera using the mouse, and activate abilities and use items via keystrokes. By default, the camera is locked to the player's chosen champion, but it is commonly accepted among the community that playing with “tunnel vision” is not conducive to victory as it prevents players from looking around the map and positioning the camera on themselves in a manner that provides the best possible strategic advantage.

The above displays a comparison of locked and unlocked cameras (Figure 2). Both images show the champion Sivir standing next to a friendly defensive tower



**Figure 2.** A comparison of locked and unlocked cameras (Riot Games, 2009).

in the top lane. The first image features a locked camera—although Sivir is in the center of the screen, her player does not have detailed vision of the potentially dangerous area ahead. The unlocked camera (also pictured) allows Sivir's player to see the enemy champions Renekton and Lucian, along with their minion wave, while remaining at a safe proximity. One of the first forms of mechanical expertise that players must acquire is the ability to continually position and reposition the game camera during play so as to maintain good vision, which will in turn allow for better strategic positioning of their champion throughout all stages of the game.

Another example of mechanical expertise is the player's knowledge of their champion's abilities and how these abilities work in combination with one another. Most champions have access to five unique abilities, each of which can be categorized as either "activated" or "passive." The former, as the name suggests, are manually activated by the player—this might be a fireball, a shield, or a teleport. Passive abilities confer a constant bonus, such as increased health regeneration or the ability to move through certain units. Many activated abilities are "skill shots" in that they are manually targeted in a direction or into a particular area, requiring mouse precision and timing for maximum effect or any effect at all. Some champions possess more than five abilities, such as transforming champions who have a set of abilities for each form, although these are few in number. A champion's abilities and how they work together (often referred to as a champion's "kit") and how easy they are to execute determine the overall difficulty of the champion. Where new players might use abilities slowly and independent of one another and struggle landing skill shots, more experienced players with stronger mechanical expertise over their champion will be able to utilize the full kit in order to defeat even enemies who have higher stats and better equipment.

One of the most popular mechanically intensive champions is Lee Sin, a highly mobile melee fighter who requires a high degree of mechanical expertise to play effectively. With creative use of his abilities, Lee Sin players can move swiftly across the battlefield and provide huge advantages for their team during skirmishes with the opposition. The mark of an adept Lee Sin player is his or her ability to make effective use of the character's ultimate ability, "Dragon's Rage," a damaging kick which shoves a single enemy champion in a chosen direction. The steps here display the professional player Thinkcard utilizing almost all of Lee Sin's kit in order to execute a perfect Dragon's Rage (Figure 3).

First, Thinkcard lands Lee Sin's skillshot ability, Sonic Wave, on the enemy Braum, controlled by Bubbadub. He then reactivates the ability to dash to Braum's position, and upon doing so immediately places a ward (a deployable item used to grant vision of the map) on the ground. ThinkCard dashes to the ward using Lee Sin's Safeguard ability, which allows him to dash to friendly champions and items. He then uses Flash, a spell accessible by all champions, to teleport behind the enemy carry (the team's main damage dealer), Caitlyn, controlled by ROBERTxLEE. Dragon's Rage is then used on Caitlyn, shoving her backward into the rest of Thinkcard's team, who easily finish her off. Even though any Lee Sin player has access to this combo, it takes practice, timing, and precision to pull off.

In addition to the process of trial and error, players will use external resources to hasten their acquisition of mechanical expertise. Given the large number of unique champions and abilities in *League of Legends*, the player base is still not aware of how certain game elements will interact if placed into contact with one another. In this light, *League of Legends* can therefore be called a game of construction rather than manipulation (Ang et al., 2010, p. 356). An interesting site of experimentation is YouTube user Holyarme Min's ongoing series *LOL Lab* (2014), which exists to

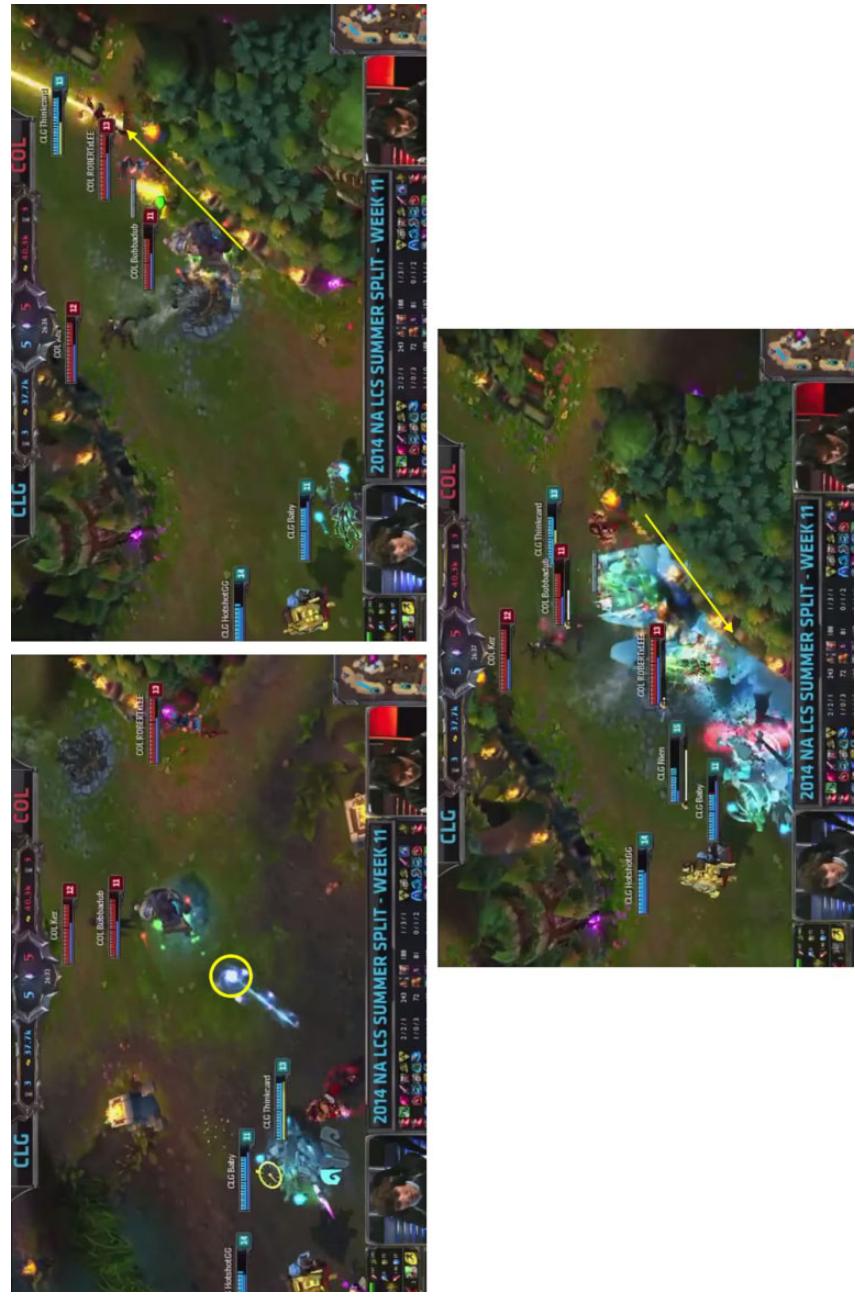


Figure 3. Thinkcard playing Lee Sin (League of Legends LCS Highlights, 2014).

answer user questions regarding uncommon ability and item interactions. In one video, a user asks what happens when the champions Sion and Malphite collide while using their ultimate abilities, both of which stipulate that the champion cannot be stopped until the ability has finished. Holyarme sets up the situation outside of a competitive context and performs the experiment, finding that Sion and Malphite simply move through each other, completely unaffected by the other's ability. Along with written guides and video tutorials, *LOL Lab* is a tool that players can use to increase their knowledge of how certain game elements interact without having to await for particular situations to arise in-game. This type of extrinsic and often collaborative activity is also popular among endgame-level players of MMORPGs—Choontanom (2012, p. 4) describes these experimental practices as ways for players to uncover the “blackboxed” mathematical rules that govern the game system.

As mentioned earlier, Chen identifies maximum level as the dividing point between mechanical and metagame expertise, as by that point, a player will generally be adept at handling the mechanical aspects of the game and in order to progress further must engage with the game's social contexts. Although it is not an MMORPG, *League of Legends* features a leveling system that, like *World of Warcraft*, determines which parts of the game a player can access. Here, players gain experience points by playing games (experience is accumulated whether a game is won or lost), and ranked mode is only unlocked once a player reaches the maximum level of 30. Until this point, players are restricted to the various unranked modes, where there are no penalties for games lost, thereby allowing new players a chance to experiment with and learn the mechanics of the game in an environment where failure is without consequence. That said, even though these modes are less “serious” than ranked, players can still gain expertise this way. In “custom” mode, a player can set up his or her own game on any of the available maps and add friends or computer-controlled bots of customizable difficulty. There is no set minimum for team size, meaning that a player can enter a game alone for the purpose of learning the map or a new champion's kit. Another popular mode is All Random, All Mid (ARAM), which takes place on the Howling Abyss, a map which consists only of a single lane. As the name suggests, players are forced to play champions chosen at random from their available pool, meaning that ARAM games are often quite unpredictable. Playing ARAM can help a player develop mechanical expertise thanks to its combination of close-quarters combat and random team compositions. The close-quarters nature of the map requires players to move efficiently and with quick reflexes so as to avoid enemy abilities, particularly those which can root the player in place, allowing easy follow-up for the rest of the enemy team. The random nature will often force the player into a champion or role he or she is unfamiliar with, and so they must learn a champion's kit over the course of the match.

Again, like *World of Warcraft*, metagame expertise is not an explicit requirement until max level, when the higher stakes tier of ranked play is unlocked and players become reliant on each other to know and understand the metagame, so they might make correct choices during both champion selection and the match itself. What

differs between the two games, however, is what the level is bound to—in *World of Warcraft*, each of the characters bound to a player have their own individual level, whereas in *League of Legends*, there is a single level tied to the player's game account. This means that once a player reaches the maximum level of 30, they will be able to compete in ranked play using any of their available characters, even those they have no experience with. However, there are 130 playable champions in *League of Legends* and it takes on average 365 games to reach maximum level, meaning that it would be unlikely for a player to garner anything other than basic mechanical expertise over each champion by the time they qualify for ranked play. This means that unlike in *World of Warcraft* and other MMORPGs, where there are far less character types (there were nine available during Chen's ethnographic study), *League of Legends* players will still be learning mechanics at max level, meaning they will have to simultaneously accumulate metagame expertise if they wish to maximize their effectiveness in ranked play.

### *League of Legends and the Metagame*

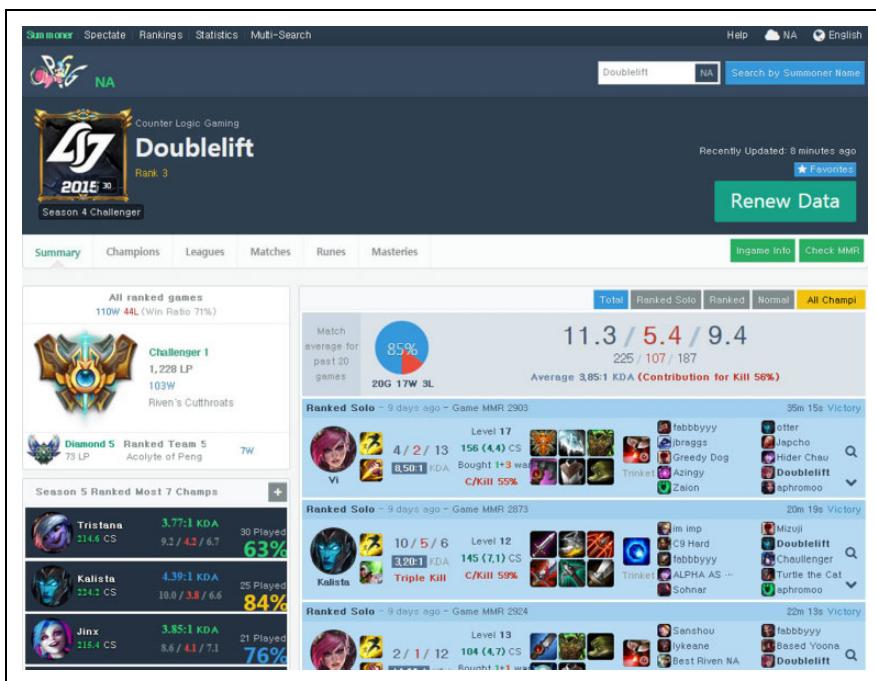
Retaining an awareness of, and being able to respond to, the constantly changing metagame is how I theorize the second stage of expertise in *League of Legends*. Andrew Garfield divides the manifestations of the metagame into four categories: what players bring to a game, what players take away from a game, what happens between games, and what happens during a game (Salen & Zimmerman, 2003, p. 482). Within the *League of Legends* community, however, the term metagame is generally used to refer to how the game is played *based on* these four categories. The metagame changes as both the game and its players change, and so what is considered the “best” approach to the game is often in a state of flux. Players will refer to a particular metagame as an “assassin metagame” or “split-push metagame” to indicate which type of champion or strategy is currently most popular.

The current *League of Legends* metagame plays an important role in determining the particular positions and roles to which champions are commonly assigned. It is therefore important, particularly in ranked play, that players are aware of this part of the metagame, so that they may participate effectively as part of a team. Position and role assignment is negotiated by players before the game begins and determines, respectively, where each player will spend most of the laning phase and what responsibilities he or she will take on during the middle and late-game phases. For example, it is common for the top lane position to be filled by a “tank” champion—one that can withstand significant damage before dying—which can, in the later portion of the game, initiate team fights and draw enemy fire. Although this framework cannot be considered a “game rule” in the formal sense, conforming to this aspect of the metagame is at times part of the “etiquette” of *League of Legends*, therefore making it somewhat of an implicit rule at times—players who do not conform to certain standards, by playing champions in roles they are not perceived to be useful in or by playing out of position, are often reported for poor behavior. Kou and Nardi

(2014, p. 8) observe that this is due to a section in the Summoner's code (a set of implicit rules created by Riot, the game developer), which states that players should always support their team (*The Summoner's Code*, 2014), and many players consider a refusal to conform to the meta as tantamount to a refusal to follow this rule. However, Riot does not feel the same way, advising that not conforming to the meta-game does not constitute a violation of the Summoner's code and as such is not a reportable offense (*Reporting a Player*, 2014). Even though "breaking the meta" is a point of contention for some players, off-meta play styles might eventually become accepted (Foo & Koivisto, 2004, p. 247)—some uncommon strategies, such as the duo-top lane and the farming-only jungle often return during metagames wherein such play styles are feasible. Understanding how the standard for play operates and how to support and/or counter off-meta strategies is therefore an important part of metagame negotiation in ranked play.

There are a number of resources and activities external to gameplay, which players access and perform in order to maximize their effectiveness as both an individual player and a team member. Players engage in round table discussion in spaces such as Reddit's *League of Legends Meta* board, where they can debate the effectiveness of a particular item or strategy using theorycrafting and other forms of analysis. Reddit, a social networking and news website, allows users to create individual forums for particular topics and can within these forums hold discussions centered on individual posts. *League of Legends Meta* serves as a place for players to engage in discussion regarding strategies as they pertain to the current metagame. Common types of post take the form of "Should I buy X or Y item on this champion?" and "Why is X champion good/bad in the current metagame?" The depth of discussion can range from short comments eschewing technical details in favor of a "common sense" approach to lengthy analysis filled with in-depth calculations. The most popular post at the time of writing is titled, "Pure Ap Fizz core build itemization or 'Math in action!'" (MGoldDragon, 2013), which profiles a number of common builds for the assassin champion Fizz and includes the results of calculations determining the most cost-efficient build paths. In the post's discussion section, a number of other players debate and offer their constructive criticism of the analysis. The existence and popularity of the *League of Legends Meta* board shows that metagaming can be a collaborative group activity and that, like *LOL Lab*, players can draw on the results of this collaboration (without necessarily participating in it) to hasten their learning.

Taylor (2012, p. 96) notes that, similar to sports and other traditional games, top-level players of online multiplayer games will endeavor to use information about their opponents to maximize their advantage. The most effective way to shut down a player in *League of Legends* is to prevent them from playing their favored champions. This can be achieved during champion selection, where players from both teams take turns banning champions (thereby prohibiting their use by any player, regardless of team) and choosing their own. When the identities of players are known, as they are in professional or local play, bans might be used as a means of countering a particular player's preferred play style, thereby forcing them into



**Figure 4.** Professional player Doublelift's statistics and match history (OP.GG, n.d.).

a champion or role they are not comfortable in playing. This is frequently the case in professional-level play, where players with limited champion pools (the number of champions with whom he or she is mechanically skilled) will find themselves consistently banned out of their comfort zone.

In online play, players are generally not aware of the names of their opponents until they are revealed on a loading screen, which appears during the period between champion select and the game itself. During this time, players might use web applications such as OP.GG (Figure 4) or LoLKing to retrieve their opponents' match histories in order to garner information on their effectiveness (or ineffectiveness) with certain champions and items. One team may even choose to focus their attack on an opponent who is on a losing streak, so as to lower their already-fractured morale. In addition to providing match histories, these websites display a player's current rank, meaning if an opponent is identified as a substantially higher rank than all the players on the team, the team might choose to alter their strategy in order to "shut down" the more experienced player. As Lynch (2013, p. 52) points out, "Playing a game is inseparable from taking part in community development," and this is particularly true for *League of Legends*—these online services accumulate and display data from a range of categories beyond individual player records. Frequency of champion

picks based on region, rank, and game type, champion win rates, and more are available for analysis by all players. Such a concept is not exclusive to *League of Legends*. Play analytics have been a core part of metagame expertise accumulation in a number of other online multiplayer games across a variety of genres, such as *Halo Wars* (Medler, 2012), *DOTA 2*, and *World of Warcraft*.

Communication also forms a part of the metagame. In solo matchmaking, teammates can communicate to one another via text chat and a ping system, which allows players to place symbols on the game map signaling danger or a vulnerable target. By default, a language filter blocks swear words and other potentially offensive terms but this can be switched off. Cross-team chat is also off by default but can be turned on, thereby allowing communication between opponents at any point during the game. Once “all-chat” is enabled, players will often engage in the type of sledging and “psyching out” that is popular among a number of traditional sports, such as cricket and baseball. Players can also mute individual teammates or opponents. Communication options beyond text and pings do not exist within the *League of Legends* game client. However, it is common for premade teams to utilize external applications, such as Skype, to communicate by voice.

The reason for the significance of the metagame in *League of Legends* and other multiplayer games can be related to the principles of ludus and paidia, proposed by Caillois (1961), which form a central part of play theory. Where ludus represents characteristics of a “game”—where there are specific rule sets and goals—paidia represents characteristics of free and explorative play. Dormans (2011, p. 2) conceptualizes ludus and paidia as opposing ends of the same spectrum, conceding that although there are games of “pure” ludus and pure paidia, most games have elements of each, and *League of Legends* is certainly no exception. In the case of *League of Legends*, this can be explained by Jensen (2013, p. 69) who also cites a spectrum of ludus and paidia, stating that “because of the transformative influences of culture, play, and a practice that has been referred to as ‘metagaming,’ paidia inevitably transforms into ludus. Similarly, ludus can also regress to transform back into paidia.” As stated earlier, the primary goal in a game instance of *League of Legends* is to destroy the enemy nexus. This along with the generic “restrictions” of video games (one cannot fly if such a thing is not permitted within the game code) are the core ludic elements of *League of Legends*. The game has, however, been available since 2009, and Bateman’s (2005) idea that when “the same group regularly return to the same playground, patterns of play will develop [and] expressions of ludus will gradually mediate the initial anarchy” certainly holds true here.

It could be theorized, then, that the game of *League of Legends* will eventually be “figured out,” but that is not so. *League of Legends*, like many other online video games, is frequently updated via patches—small pieces of mandatory downloadable content, which can add and remove content or alter preexisting code. Generally, *League of Legends* players see a new patch every 2 or 3 weeks during the competitive season, with a single massive overhaul each year. Most of the changes in each patch are piloted in the “public beta environment (PBE),” Riot Games’ test server for

*League of Legends*. Players must apply to play on the beta server and testers are cycled out every few weeks. PBE[ testers can provide feedback to changes and notify the developers of bugs via the PBE forum, the content of which is publically accessible (<http://boards.pbe.leagueoflegends.com/en/>). Changes made in the PBE are often reversed or changed further before their official implementation in the main servers. We can therefore refer to *League of Legends* as a game of “multiple incarnations” (Newman, 2012, p. 136), and that the *League of Legends* being played at this moment by millions of players is not the same *League of Legends* that existed just 2 months ago. DeKoven (2005, p. 520) states that the reason behind changing a game is to “restore equilibrium” between the “playing mind” and the “gaming mind”—that is, *paidia* and *ludus*, respectively. For a game to be considered interesting and enjoyable, according to DeKoven, there needs to be some level of *paidia*—some level of experimentation—present. Once the players have perfected the game or are at least aware of the most effective strategy, gameplay becomes stagnant. A common example of this is the inadvertent overpowering of a particular champion that might come with one particular patch—the players pick up on the oppressive nature of the champion, and he or she obtains 100% pick/ban status—that is, in professional competition, the champion is either picked or banned in every single game of the particular patch. Generally, a subsequent patch will contain a nerf (a de-powering) of whatever it is that causes the champion to be so powerful, whether that is an aspect of the champion itself, an item, or the game environment. Although most patches consist only of changes to existing game content, new champions, new items, and item removals will feature infrequently and tend to have a much larger impact on the overall metagame.

## Conclusions and Further Study

As with a number of other multiplayer games, expertise in *League of Legends* is composed of binary elements. The first is mechanical expertise, which relates to in-game elements such as interface navigation and avatar control, some of which might carry over from previous play experiences. The second form of expertise is metagame expertise, which is the awareness of and ability to negotiate the game around the game: it could be the formulation of new strategies after a patch, the use of mathematical techniques to determine the effectiveness of a particular item or ability combination, or the analysis of data sets for the purposes of improving one’s in-game effectiveness. It could perhaps be theorized that without this second layer of expertise, the player will find himself or herself at a disadvantage when playing against those who are more familiar with the state of the metagame, even if they have comparative mechanical expertise. This is certainly the belief of the player base at large, as it is common for toxic behavior to arise in champion select when one player demonstrates a lack of metagame expertise by choosing an out-of-meta champion, role, or position, as it assumed that this choice will place the team at an immediate disadvantage.

The *League of Legends* metagame and the process of metagaming is such a major part of competitive play that it certainly warrants further study—while it may not be required to participate, it is certainly required to be effective in the competitive environment, especially at higher levels. Here, I will outline areas in which I feel there to be opportunities for increased scholarly attention.

First, the social conventions which value or devalue player actions in and around play are potentially significant areas of study. Alluded to in this article is a distinction between conventional and unconventional forms of play—in my experience, I have found that in competitive play, conforming to the conventions of the metagame is an implicit part of behavioral etiquette, and breaking the meta can often be met by rejection from allied players. Even though Riot state clearly that a player cannot be punished for playing an off-meta champion, position or role, or purchasing an off-meta item “build,” players are still expected to conform to what is collectively considered the best way to play. A qualitative or quantitative (statistical) inquiry into the social response to unconventional play forms, like that of Toh’s (2014) on *Lord of the Rings* online, might be useful in order to formulate an accurate portrait of *League of Legends* players and their attitudes toward deviant play forms in the competitive arena. The styles of celebrity players such as Disco Heat, who intentionally uses off-meta champions as a way of showing the oftentimes hostile attitudes attracted by those who do not play by “the rules,” and Annie Bot, a high-level player famous for playing only a single champion, are also worthwhile points of analysis. Such a study might also be used to determine the genuine worth of metagame expertise, compared to mechanical expertise, at various levels of competition. It would certainly be interesting to find how far a mechanically strong *League of Legends* player could climb in the rankings while adopting *only* out-of-meta play styles.

Also mentioned here is the unique overlap of mechanical and metagame expertise accumulation in *League of Legends*. It would therefore be useful to look at nascent forms of play enacted by those new to *League of Legends* and how these players and their play styles evolve as they simultaneously garner both mechanical and metagame expertise. There are preexisting studies of how players negotiate metagames, such as Harper’s (2010) ethnographic account of social gameplay at a fighting game tournament, though it would be interesting to chart a new player’s progress within a metagame as prevalent and (in some areas) prescriptive as that of *League of Legends*.

Third, observable via online stat-tracking services is, throughout all levels of play, a stark difference between the metagames of team play (i.e., when teams are premade) and solo play (when teams are made up of five unrelated players). It is apparent that when a premade team plays a game on Summoner’s Rift, their play styles and champion choices are greatly influenced by the ability to coordinate their movements and team fights on account of improved communication. Unless they utilize third-party software (such as Skype), solo players may only communicate via pings and text chat, making coordination more difficult. With this being the case, certain champion picks that are priorities in solo play do not receive equal attention

in team play and vice versa. This is somewhat unique to the MOBA genre. An investigation into the relationship between the metagame and player communication would be worthwhile.

Whether it be the ability to analyze opponents, discuss strategy, and negotiate implicit rules, metagame expertise is a fundamental component of competitive effectiveness not only for *League of Legends* but also for similar strategically deep and constantly evolving, multiplayer video games. Other MOBAs, such as *DOTA 2* and *Heroes of Newerth*, sport their own deep and complex metagames, as does real-time strategy game *Starcraft 2* and the first person shooter *Counter-Strike*. Mastery of the mechanical aspects of these games is not always enough to guarantee competitive success—if a player is to maximize his or her chances of victory in ranked play, metagame expertise must be acquired.

### **Declaration of Conflicting Interests**

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

### **Funding**

The author(s) received no financial support for the research, authorship, and/or publication of this article.

### **References**

- Ang, C. S., Wildon, S., & Zaphiris, P. (2010). Computer games and sociocultural play: An activity theoretical perspective. *Games and Culture*, 5, 354–380. Retrieved from <http://dx.doi.org/10.1177/1555412009360411>
- Armitage, G. (2003). An experimental estimation of latency sensitivity in multiplayer Quake 3. In *The 11th IEEE International Conference on Networks, 2003. ICON2003* (pp. 137–141). Retrieved from <http://doi.org/10.1109/ICON.2003.1266180>
- Bateman, C. (2005, December 23). *The Anarchy of Paidia* [Blog post]. Retrieved from [http://onlyagame.typepad.com/only\\_a\\_game/2005/12/the\\_anarchy\\_of\\_1.html](http://onlyagame.typepad.com/only_a_game/2005/12/the_anarchy_of_1.html)
- Caillois, R. (1961). *Man, play, and games*. Chicago: University of Illinois Press.
- Chen, M. (2010). *Leet Noobs: Expertise and collaboration in a world of warcraft player group as distributed sociomaterial practice* (Doctoral dissertation, University of Washington). Retrieved from <https://digital.lib.washington.edu/researchworks/handle/1773/16275>
- Choontanom, T. (2012). *The manifestation of player discipline in video games* (Master's Thesis). Retrieved from ProQuest Dissertations and Theses (103557508).
- DeKoven, B. (2005). Changing the game. In K. Salen & E. Zimmerman (Eds.), *The game design reader: A rules of play anthology* (pp. 518–537). Cambridge, MA: The MIT Press.
- Dormans, J. (2011). Integrating emergence and progression. In *Proceedings of DiGRA 2011 Conference: Think design play*. Retrieved from <http://www.digra.org/digital-library/forums/6-think-design-play/>

- OP.GG. (n.d.). *Doublelift*. Retrieved April 7, 2015, from <https://na.op.gg/summoner/userName=Doublelift>
- Foo, C. Y., & Koivisto, E. M. I. (2004). Defining grief play in MMORPGs: Player and developer perceptions. In *Proceedings of the 2004 ACM SIGCHI International Conference on Advances in computer entertainment technology* (pp. 245–250). Retrieved from <http://dx.doi.org/10.1145/1067343>
- Harper, T. L. (2010). *The art of war: Fighting games, performativity, and social game play* (Doctoral dissertation, Ohio University). Retrieved from [https://etd.ohiolink.edu/ap/10?0::NO:10:P10\\_ACCESSION\\_NUM:ohiou1283180978](https://etd.ohiolink.edu/ap/10?0::NO:10:P10_ACCESSION_NUM:ohiou1283180978)
- Henderson, T. (2001). Latency and user behaviour on a multiplayer game server. In J. Crowcroft & M. Hofmann (Eds.), *Networked group communication* (pp. 1–13). Berlin, Germany: Springer. Retrieved from [http://link.springer.com/chapter/10.1007/3-540-45546-9\\_1](http://link.springer.com/chapter/10.1007/3-540-45546-9_1)
- Holyarmer, M. (2014, October 15). *Holyarmer's LOL Lab #30—Sion featured* [Video file]. Retrieved from <https://www.youtube.com/watch?v=kMj7KhKJ4lY>
- Humphreys, A., & de Zwart, M. (2012). Griefing, massacres, discrimination, and art: The limits of overlapping rule sets in online games. *UC Irvine Law Review*, 2, 507–536. Retrieved from <http://hdl.handle.net/2440/77704>
- Jakobsson, P., Pargman, D., & Rambusch, J. (2007). Exploring E-sports: A case study of game-play in counter-strike. In *Proceedings of the 2007 DiGRA International Conference: Situated Play*. Retrieved from <http://www.digra.org/digital-library/forums/4-situated-play/>
- Jansz, J., & Martens, L. (2005). Gaming at a LAN event: The social context of playing video games. *New Media & Society*, 7, 333–355. Retrieved from <http://nms.sagepub.com/content/7/3/333.abstract>
- Jensen, G. H. (2013). Making sense of play in video games: Ludus, Paidia, and possibility spaces. *Eludamos*, 7, 69–80. Retrieved from <http://www.eludamos.org/eludamos/index.php/eludamos/article/viewArticle/vol7no1-4/7-1-4-html>
- Kou, Y., & Gui, X. (2014). Playing with strangers: Understanding temporary teams in League of Legends. In *CHI Play '14: Proceedings of the first ACM SIGCHI annual symposium on computer-human interaction in play* (pp. 161–169). Retrieved from <http://dx.doi.org/10.1145/2658537.2658538>
- Kou, Y., & Nardi, B. (2014). Governance in League of Legends: A Hybrid System. In *Proceedings of the 9th International Conference on the Foundations of Digital Games*. Retrieved from <http://www.fdg2014.org/proceedings.html>
- League of Legends LCS Highlights. (2014, August 3). *CLG thinkcard amazing leesin insec kick—CLG vs COL—NA LCS summer split 2014* [Video file]. Retrieved from <https://www.youtube.com/watch?v=A2JQFyQYNw8>
- Lynch, H. L. (2013). *Composing a gamer: A case study of one gamer's experience of symbiotic flow* (Doctoral dissertation, Georgia State University). Retrieved from [http://scholarworks.gsu.edu/msit\\_diss/112/](http://scholarworks.gsu.edu/msit_diss/112/)
- McArthur, V., Peyton, T., Jenson, J., Taylor, N., & de Castell, S. (2012). Knowing, not doing: Modalities of gameplay expertise in world of warcraft addons. In A. Williams & E. Robles (Eds.), *CHI '12 extended abstracts on human factors in computing systems (CHI EA '12)* (pp. 101–110). New York, NY: ACM.

- Medler, B. (2012). *Play with data—An exploration of play analytics and its effect on player experiences* (Doctoral dissertation, Georgia Institute of Technology). Retrieved from <http://hdl.handle.net/1853/44888>
- MGoldDragon. (2013, August 11). *Pure AP Fizz core build itemization OR “Math in action!”* [Forum post]. Retrieved from [http://www.reddit.com/r/LeagueofLegendsMeta/comments/1k63l5/pure\\_ap\\_fizz\\_core\\_build\\_itemization\\_or\\_math\\_in/](http://www.reddit.com/r/LeagueofLegendsMeta/comments/1k63l5/pure_ap_fizz_core_build_itemization_or_math_in/)
- Neuenschwander, B. (2008). Playing by the rules: Instruction and acculturation in role-playing games. *E-Learning and Digital Media*, 5, 189–198. Retrieved from <http://dx.doi.org/10.2304/elea.2008.5.2.189>
- Newman, J. (2012). Ports and patches: Digital games as unstable objects. *Convergence*, 18, 135–142. Retrieved from <http://dx.doi.org/10.1177/1354856511433688>
- Przybylski, A. K., Rigby, C. S., & Ryan, R. M. (2010). A motivational model of video game engagement. *Review of General Psychology*, 14, 154–166. Retrieved from <http://dx.doi.org/10.1037/a0019440>
- Ratan, A. R., Taylor, N., Hogan, J., Kennedy, T., & Williams, D. (2015). Stand by your man: An examination of gender disparity in *League of Legends*. *Games and Culture*, 10, 438–462.
- Reeves, S., Brown, B., & Laurier, E. (2009). Experts at play: Understanding skilled expertise. *Games and Culture*, 4, 205–227. Retrieved from <http://dx.doi.org/10.1177/1555412009339730>
- Reporting a Player. (2014). Retrieved January 15, 2015, from <https://support.riotgames.com/hc/en-us/articles/201752884-Reporting-a-Player>
- Riot Games. (2014). *Our games*. Retrieved January 15, 2015, from <http://www.riotgames.com/our-games>
- Riot Games. (2009). *League of legends* [PC game]. Santa Monica, CA: Riot Games.
- Salen, K., & Zimmerman, E. (2003). *Rules of play: Game design fundamentals*. Cambridge, MA: MIT Press.
- Summoner's Rift [Image]. (2014). Retrieved from <http://na.leagueoflegends.com/en/news/game-updates/features/dev-blog-finding-your-place-rift>
- Taylor, N., de Castell, S., Jenson, J., & Humphrey, M. (2011). Modeling play: Re-casting expertise in MMOGs. In S. N. Spencer (Ed.), *ACM SIGGRAPH 2011 Game Papers (SIGGRAPH '11)* (pp. 49–54). New York, NY: ACM.
- Taylor, T. L. (2012). *Raising the stakes*. Cambridge, MA: MIT Press.
- The Summoner's Code. (2014). Retrieved January 15, 2015, from <http://gameinfo.na.leagueoflegends.com/en/game-info/get-started/summoners-code/>
- Toh, W. (2014). An analysis of open world PvP in LOTRO'S PvMP as a case study for PvP games. *Press Start*, 1, 37–58. Retrieved from <http://press-start.gla.ac.uk/index.php/press-start/issue/view/1>

## Author Biography

**Scott Donaldson** is a PhD student of Media, Culture and Creative Arts at Curtin University, Australia. His thesis investigates the significance of metagame negotiation in professional and high-level solo play in *League of Legends*.

## TOWARDS OPTIMIZING ENTERTAINMENT IN COMPUTER GAMES

**Georgios N. Yannakakis** □ *Mærsk Mc-Kinney Møller Institute, University of Southern Denmark, Odense M, Denmark*

**John Hallam** □ *Mærsk Mc-Kinney Møller Institute, University of Southern Denmark, Odense M, Denmark*

□ *Mainly motivated by the current lack of a qualitative and quantitative entertainment formulation of computer games and the procedures to generate it, this article covers the following issues: It presents the features—extracted primarily from the opponent behavior—that make a predator/prey game appealing; provides the qualitative and quantitative means for measuring player entertainment in real time, and introduces a successful methodology for obtaining games of high satisfaction. This methodology is based on online (during play) learning opponents who demonstrate cooperative action. By testing the game against humans, we confirm our hypothesis that the proposed entertainment measure is consistent with the judgment of human players. As far as learning in real time against human players is concerned, results suggest that longer games are required for humans to notice some sort of change in their entertainment.*

Intelligent interactive opponents can provide more enjoyment to a vast gaming community of constant demand for more realistic, challenging, and meaningful entertainment (Fogel et al. 2004; Champandard 2004). However, given the current state-of-the-art in artificial intelligence (AI) in computer games, it is unclear which features of any game contribute to the satisfaction of its players, and thus it is also uncertain how to develop enjoyable games. Because of this lack of knowledge, most commercial and academic research in this area is fundamentally incomplete. The challenges we consider in this article are to provide qualitative and quantitative means for distinguishing a game's enjoyment value and to develop efficient AI tools to automatically generate entertainment for the player.

In our previous work (Yannakakis and Hallam 2004), we defined criteria that contribute to the satisfaction for the player, which map to

The authors would like to thank the anonymous reviewers for their valuable feedback and the players for their vital contribution to this work.

Address correspondence to G. N. Yannakakis, Mærsk Mc-Kinney Møller Institute, University of Southern Denmark, Odense M, Denmark. E-mail: georgios@mmti.sdu.dk

characteristics of the opponent behavior in computer games. According to our hypothesis, the player-opponent interaction—rather than the audiovisual features, the context, or the genre of the game—is the property that primarily contributes the majority of the quality features of entertainment in a computer game. Based on this fundamental assumption, we introduced a metric for measuring the real-time entertainment value of predator/prey games.

According to our second hypothesis, entertainment is generated when adaptive learning procedures occur in real time. This allows for opponents to learn while playing against the player and adapt with regards to his/her strategy. When such a mechanism is built upon cooperative opponents, it is more likely that the game's interest value improves drastically. Using the Pac-Man game as a test-bed (Yannakakis and Hallam 2004a, 2005) and focusing on the non-player characters' (NPC's) cooperative behavior, a robust online (i.e., while the game is played) neuro-evolution (i.e., automatic shaping of artificial neural networks by the use of artificial evolution) learning mechanism was presented, which was capable of increasing the game's interest as well as keeping that interest at high levels while the game was being played. This mechanism demonstrated high robustness and adaptability to changing hand-crafted player strategies in a relatively simple playing stage. Additional experiments in a predator/prey game of a more abstract design called Dead End (Yannakakis and Hallam 2004b) displayed the effectiveness of the proposed methodology in dissimilar games of the same genre and expanded the applicability of the method.

In the work presented here, apart from experiments with computer-guided players, human players are used for testing the Pac-Man game in a more complex stage. The main objective of this work is to establish the interest measure proposed as an efficient generic predator/prey game metric, by the approval of human judgment. In other words, we attempt to cross-correlate the human notion of interest to the proposed interest metric in one of the most representative test-beds of this computer games domain. Furthermore, we examine the online learning algorithm's abilities against humans and attempt to discover whether it maintains its robustness and adaptability under real conditions, that is, against human players.

## **LEARNING IN GAMES**

The majority of research on learning in games is built on board or card games. In the last decade, many researchers have been involved in the development of intelligent opponents in board or card games. Some of the attempts include evolutionary learning approaches applied, from tic-tac-toe (Fogel 1993), to checkers (Fogel [2002] among others) and Go (Richards et al. 1998). In Tesauro (2002), a temporal difference learning

mechanism generates computer opponents capable of beating even expert humans in backgammon. These games are board games simulated in computers and therefore sometimes people refer to them as “computer games.” However, when we refer to computer games, we refer to the category of commercial games played by NPCs in virtual worlds.

Based on the success of this mentioned research on board games, increasing computing power and the commercial possibilities of computer games, very recently, researchers have attempted to introduce AI into computer games and have discussed the theoretical perspective of learning in different categories of games. Laird (2002) surveys the state of research in using AI techniques in interactive computer games. He also provides a taxonomy of games and the importance of computer games as experimental environments for strong AI application. Furthermore, Isla and Blumberg (2002) suggest potential research directions in AI game development, emphasizing to the emotional state and the perceived information of the character. Taylor (2000) attempts to bridge the gap between game development and modern AI by proposing artificial life techniques for generating physically modeled characters.

Game AI researchers, in their majority, focus on the genre of first-person shooter (FPS) and real-time strategy (RTS) games, primarily because of their popularity and secondarily because of their open-source game engines. Alex J. Champandard (2004) is using an FPS game to propose and apply a plethora of forms of AI techniques (varying from simple scripting to adaptive learning) for specific tasks like movement, shooting, and weapon selection. Khoo (2002) developed an inexpensive AI technique based on the well-known Eliza program (Weizenbaum 1966), so that users get the impression of playing against humans instead of bots. In Cole et al. (2004), the parameters of the *Counter-Strike* built-in weapon selection rules are tuned by using artificial evolution. Furthermore, there have been attempts to mimic human behavior offline, from samples of human playing, in a specific virtual environment. Alternatively, dynamic scripting and evolutionary learning has been used in a real-time strategy (RTS) games (Ponsen and Spronck 2004). In Thurau et al. (2004), among others, human-like opponent behaviors are emerged through supervised-learning techniques in *Quake*. Even though complex opponent behaviors emerge, there is no further analysis of whether these behaviors contribute to the satisfaction of the player (i.e., interest of game). In other words, researchers hypothesize—by observing the vast number of multi-player online games played daily on the Web, for example—that by generating human-like opponents, they enable the player to gain more satisfaction from the game. This hypothesis might be true up to a point; however, since there is no explicit notion of interest defined, there is no evidence that a specific opponent behavior generates more or less interesting games. Such a

hypothesis is the core of Iida's work on board games. He proposed a general metric of entertainment for variants of chess games depending on average game length and possible moves (Iida et al. 2003).

## Identifying and Augmenting Entertainment

There have been several psychological studies to identify what is “fun” in a game and what engages people playing computer games. Theoretical approaches include Malone's (1981) principles of intrinsic qualitative factors for engaging game play, namely, challenge, curiosity, and fantasy as well as the well-known concepts of the theory of flow (Csikszentmihalyi 1990) incorporated in computer games as a model for evaluating player enjoyment, namely, *Game Flow* (Sweetser and Wyeth 2005). A comprehensive review of the literature on qualitative approaches for modeling player enjoyment demonstrates a tendency of overlapping with Malone's and Csikszentmihalyi's foundational concepts. Many of these approaches are based on Lazzaro's “fun” clustering, which uses four entertainment factors based on facial expressions and data obtained from game surveys on players (Lazzaro 2004): hard fun, easy fun, altered states, and socialization. Koster's (2005) theory of fun, which is primarily inspired by Lazzaro's four factors, defines “fun” as the act of mastering the game mentally. An alternative approach to fun capture is presented in Read et al. (2002), where fun is composed of three dimensions: durability, engagement, and expectations. Questionnaire tools and methodologies are proposed in order to empirically capture the level of fun for evaluating the usability of novel interfaces with children.

Work in the field of quantitative entertainment capture and augmentation is based on the hypothesis that the player-opponent interaction—rather than the audiovisual features, the context, or the genre of the game—is the property that contributes to the majority of the quality features of entertainment in a computer game (Yannakakis and Hallam 2004a). Based on this fundamental assumption, a metric for measuring the real-time entertainment value of predator/prey games was designed, and established as a generic interest metric for prey/predator games (Yannakakis and Hallam 2005a; 2005b). Further studies by Yannakakis and Hallam (2006) have shown that artificial neural networks (ANN) and fuzzy neural networks can extract a better estimator of player satisfaction than a human-designed one, given appropriate estimators of the challenge and curiosity of the game (Malone 1981) and data on human players' preferences. Similar work in adjusting a game's difficulty include endeavors through reinforcement learning (Andrade et al. 2005), genetic algorithms (Verma and McOwan 2005), probabilistic models (Hunicke and Chapman 2004), and dynamic scripting (Spronck et al. 2004). However, the aforementioned attempts

are based on the assumption that challenge is the only factor that contributes to enjoyable gaming experiences while results reported have not been cross-verified by human players.

A step further to entertainment capture is towards games of richer human-computer interaction and affect recognizers, which are able to identify correlations between physiological signals and the human notion of entertainment. Experiments by Yannakakis et al. (2006) have already shown a significant effect of the average heart rate of children's perceived entertainment in action games played in interactive physical playgrounds. Moreover, Rani et al. (2005) propose a methodology for detecting the anxiety level of the player and appropriately adjusting the level of challenge in the game of "Pong." Physiological state (heart-rate, galvanic skin response) prediction models have also been proposed for potential entertainment augmentation in computer games (McQuiggan et al. 2006).

We choose predator/prey games as the initial genre of our game research since, given our aims, they provide us with unique properties. In such games, we can deliberately abstract the environment and concentrate on the characters' behavior. The examined behavior is cooperative since cooperation is a prerequisite for effective hunting behaviors. Furthermore, we are able to easily control a learning process through online interaction. In other words, predator/prey games offer a well-suited arena for initial steps in studying cooperative behaviors generated by interactive online learning mechanisms. Even though other genres of games (e.g., FPS games) offer similar properties, researchers have not yet focused on the novel directions of human-verified entertainment capture and augmentation that are presented in this article.

## PREDATOR/PREY GAMES

Predator/prey games have been a very popular category of computer games and among its best representatives is the classical Pac-Man game released by Namco (Japan) in 1980. Even though Pac-Man's basic concept—the player's (*PacMan's*) goal is to eat all the pellets appearing in a maze-shaped stage while avoiding being killed by four opponent characters named "*Ghosts*"—and graphics are very simple, the game still keeps players interested after so many years, and its basic ideas are still found in many newly released games.

Kaiser et al. (1998) attempted to analyze emotional episodes, facial expressions, and feelings of humans playing a predator/prey computer game similar to Pac-Man (Kaiser and Wehrle 1996). Other examples in the Pac-Man domain literature include researchers attempting to teach a controller to drive *Pac-Man* in order to acquire as many pellets as possible and to avoid being eaten by *Ghosts*. Koza (1992) considers the problem of

controlling an agent in a dynamic nondeterministic environment and, therefore, sees Pac-Man as an interesting multi-agent environment for applying offline learning techniques based on genetic programming. Other approaches, such as incremental learning (Gallagher and Ryan 2003) and neuro-evolution (Lucas 2005), have also been applied for producing effective Pac-Man playing strategies. The same Pac-Man application domain has been used for analyzing size and generality issues in genetic programming (Rosca 1996).

On the other hand, there are many researchers who use predator/prey domains in order to obtain efficient emergent teamwork of either homogeneous or heterogeneous groups of predators. For example, Luke and Spector (1996), among others, have designed an environment similar to the Pac-Man game (the Serengeti world), in order to examine different breeding strategies and coordination mechanisms for the predators. Finally, there are examples of work in which both the predators' and the prey's strategies are co-evolved in continuous or grid-based environments (Haynes and Sen 1995; Miller and Cliff 1994).

Similar to Luke and Spector (1996), we view the domain from the predators' perspective and we attempt to emerge effective hunting teamwork offline based on evolutionary computation techniques applied to homogeneous neural controlled (Yao 1999) predators. However, playing a predator/prey computer game like Pac-Man against optimal hunters cannot be interesting because the player is consistently and effectively killed.

Researchers have generally shown that online learning in computer games is feasible through careful design and effective learning methodologies. On that basis, Yannakakis et al. ( 2004a; 2004c) introduced a neuro-evolution mechanism that acts in real time that optimizes each opponent individually (heterogeneous game environment) for generating appealing games rather than high opponent performance (Demasi and Cruz 2002; Graepel et al. 2004).

## **INTERESTING BEHAVIOR**

As noted, predator/prey games will be our test-bed genre for the investigation of enjoyable games. More specifically, in the games studied, the prey is controlled by the player and the predators are the computer-controlled opponents (nonplayer characters, or NPCs).

In the approach presented in this section, a quantitative metric of player satisfaction is designed based on general criteria of enjoyment. The first step towards generating enjoyable computer games is therefore to identify the criteria or features of games that collectively produce enjoyment (or else interest) in such games. Second, quantitative estimators for these criteria are defined and combined, in a suitable mathematical

formula, to give a single quantity correlated with player satisfaction (interest). Finally, this formulation player interest is tested against human players' judgment in real conditions using the Pac-Man test-bed.

Following the principles of Yannakakis and Hallam (2004a), we will ignore mechanics, audiovisual representation, control, and interface contributions to the enjoyment of the player and we will concentrate on the opponents' behaviors. A well-designed and popular game such as Pac-Man can fulfil all aspects of player satisfaction incorporated in the mentioned design game features. The player, however, may contribute to his/her entertainment through interaction with the opponents of the game and, therefore, it is implicitly included in the interest formulation presented here, see also Yannakakis and Maragoudakis (2005), for studies of the player's impact on his/her entertainment.

## Criteria

By observing the opponents' behavior in various predator/prey games, we attempted to identify the key features that generate entertainment for the player. These features were experimentally cross-validated against various opponents of different strategies and redefined when appropriate. According to Yannakakis and Hallam (2004a) and (2005b) the criteria that collectively define interest on any predator/prey game are briefly as follows.

1. Appropriate level of challenge (*when the game is neither too hard nor too easy*).
2. Behavior diversity (*when there is diversity in opponents' behavior over the games*).
3. Spatial diversity (*when opponents' behavior is aggressive rather than static*). That is, predators that move constantly all over the game world and cover it uniformly. This behavior gives the player the impression of an intelligent strategic opponents' plan, which increases the game interest.

## Metrics

In order to estimate and quantify each of the three aforementioned interest criteria, we let the examined group of opponents play the game  $N$  times (each game for a sufficiently large evaluation period of  $t_{\max}$  simulation steps) and we record the simulation steps  $t_k$  taken to kill the player, as well as the total number of the opponents' visits  $v_{ik}$  at each cell  $i$  of the grid game field for each game  $k$ . In the case where the game's motion is continuous, a discretization of the field's plane up to the character's size can serve this purpose.

Given these, the quantifications of the three interest criteria proposed above can be presented as follows.

1. *Appropriate Level of Challenge:* According to the first criterion, an estimate of how interesting the behavior is, is given by  $T$  in (1).

$$T = [1 - (E\{t_k\} / \max\{t_k\})]^{p_1}, \quad (1)$$

where  $E\{t_k\}$  is the average number of simulation steps taken to kill the prey-player over the  $N$  games;  $\max\{t_k\}$  is the maximum  $t_k$  over the  $N$  games; and  $p_1$  is a weighting parameter.

$p_1$  is adjusted so as to control the impact of the bracketed term in the formula for  $T$ . By selecting values of  $p_1 < 1$ , we reward quite challenging opponents more than near-optimal killers, since we compress the  $T$  value toward 1. More details on the adjustment of the  $p_1$  value for the Pac-Man game will follow.

The  $T$  estimate of interest demonstrates that the greater the difference between the average and the maximum number of steps taken to kill the player, the higher the interest of the game. Given (1), both easy killing (“*too easy*”) and near-optimal (“*too hard*”) behaviors receive low interest estimate values (i.e.,  $E\{t_k\} \simeq \max\{t_k\}$ ). This metric is also called “challenge.”

2. *Behavior Diversity:* The interest estimate for the second criterion is given by  $S$  in (2).

$$S = (\sigma/\sigma_{max})^{p_2}, \quad (2)$$

where

$$\sigma_{max} = \frac{1}{2} \sqrt{\frac{N}{(N-1)} (t_{max} - t_{min})} \quad (3)$$

and  $\sigma$  is the standard deviation of  $t_k$  over the  $N$  games;  $\sigma_{max}$  is an estimate of the maximum value of  $\sigma$ ;  $t_{min}$  is the minimum number of simulation steps required for predators to kill the prey obtained by playing against some “well”-behaved fixed strategy near-optimal predators ( $t_{min} \leq t_k$ ); and  $p_2$  is a weighting parameter.

The  $S$  increases proportionally with the standard deviation of the steps taken to kill the player over  $N$  games. Therefore, using  $S$  as defined here, we promote predators that produce high diversity in the time taken to kill the prey.

3. *Spatial Diversity:* A good measure for quantifying the third interest criterion is through entropy of the predators’ cell visits in a game, which

quantifies the completeness and uniformity with which the opponents cover the stage. Hence, for each game, the cell visit entropy is calculated and normalized into [0, 1] via (4).

$$H_n = \left[ -\frac{1}{\log V_n} \sum_i \frac{v_{in}}{V_n} \log \left( \frac{v_{in}}{V_n} \right) \right]^{p_3}, \quad (4)$$

where  $V_n$  is the total number of visits of all visited cells (i.e.,  $V_n = \sum_i v_{in}$ ) and  $p_3$  is a weighting parameter.  $p_3$  is adjusted in order to promote very high  $H_n$  values and furthermore to emphasize the distinction between high and low normalized entropy values. Appropriate  $p_3$  parameter values, which serve this purpose, are those greater than one ( $p_3 = 4$  in this article), since they stretch the value of  $H_n$  away from 1.

Given the normalized entropy values  $H_n$  for all  $N$  games, the interest estimate for the third criterion can be represented by their average value  $E\{H_n\}$  over the  $N$  games. This implies that the higher the average entropy value, the more interesting the game is.

The three individual criterion metrics defined are combined linearly to produce a single metric of interest (Equation 5) whose properties match the qualitative considerations developed.

$$I = \frac{\gamma T + \delta S + \epsilon E\{H_n\}}{\gamma + \delta + \epsilon}, \quad (5)$$

where  $I$  is the interest value of the predator/prey game;  $\gamma$ ,  $\delta$  and  $\epsilon$  are criterion weight parameters.

The interest metric introduced in (5) can be applied effectively to any predator/prey computer game, because it is based on generic features of this category of games (see Yannakakis and Hallam [2004b; 2005] for successful applications to dissimilar predator/prey games). These features include the time required to kill the prey and the predators' entropy throughout the game field. We therefore believe that (5)—or a similar measure of the same concepts—constitutes a generic interest approximation of predator/prey computer games. Moreover, given the two first interest criteria previously defined, the approach's generality is expandable to all computer games. Indeed, no player likes any computer game that is too difficult or too easy to play and, furthermore, any player would enjoy diversity throughout the play of any game. The third interest criterion is applicable to games where spatial diversity is important which, apart from predator/prey games, may also include action, strategy, and team sports games

according to the computer game genre classification of Laird and van Lent (2000).

The approach to entertainment modeling represented by equation (5) is both innovative and efficient. However, it should be clear that there are many possible formulae, such as equation (5), which would be consistent with the qualitative criteria proposed for predator/prey games. Other successful quantitative metrics for the appropriate level of challenge, the opponents' diversity and the opponents' spatial diversity may be designed and more qualitative criteria may be inserted in the interest formula. Alternative mathematical functions for combining and weighting the various criteria could be employed.

For example, other metrics for measuring the appropriate level of challenge could be used: One could come up with a  $T$  metric assuming that the appropriate level of challenge follows a Gaussian distribution over  $E\{t_k\}$  and that the interest value of a given game varies, depending on how long it is—*very short* ( $E\{t_k\} \approx t_{min}$ ) games tend to be frustrating and *long games* ( $E\{t_k\} \approx \max\{t_k\}$ ) tend to be boring. (However, very short games are not frequent in the experiments presented here and, therefore, by varying the weight parameter  $p_1$  in the proposed  $T$  metric [see (1)], we are able to obtain an adequate level of variation in measured challenge.)

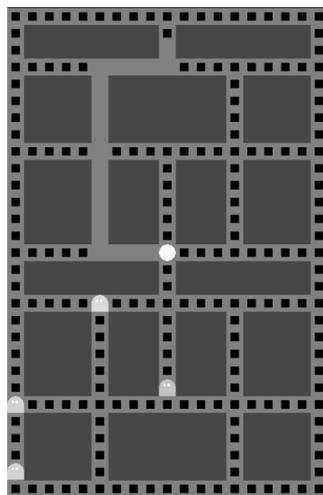
The question remains, however, whether the number produced by such a formula really captures anything useful concerning a notion so potentially complex as human enjoyment. That question is addressed next.

## THE PAC-MAN TEST-BED

The computer game test-bed studied is a modified version of the original Pac-Man computer game released by Namco. The player's (*Pac-Man's*) goal is to eat all the pellets appearing in a maze-shaped stage, while avoiding being killed by the four *Ghosts*. The game is over when either all pellets in the stage are eaten by *Pac-Man*, *Ghosts* manage to kill *Pac-Man*, or a pre-determined number of simulation steps is reached without any of these occurring. In that case, the game restarts from the same initial positions for all five characters.

Compared to commercial versions of the game, a number of features (e.g., power pills) are omitted for simplicity; these features do not qualitatively alter the nature of “interesting” in games of low interest. Cross-validation of this statement appears through the judgment and the beliefs of human players of both the original and this version of the game.

As stressed previously, the Pac-Man game is investigated from the viewpoint of *Ghosts* and more specifically how *Ghosts'* emergent adaptive behaviors can contribute to the interest of the game. Pac-Man—as a computer game domain for emerging adaptive behaviors—is a two-dimensional,



**FIGURE 1** Snapshot of the Pac-Man game.

multi-agent, grid-motion, predator/prey game. The game field (i.e., stage) consists of corridors and walls. Both the stage's dimensions and its maze structure are predefined. For the experiments presented in this article, we use a  $19 \times 29$  grid maze-stage where corridors are one grid-cell wide (see Figure 1).

The characters visualized in the Pac-Man game (as illustrated in Figure 1) are a white circle that represents *Pac-Man* and four ghost-like characters representing the *Ghosts*. Being *Ghosts*, one of their properties is permeability, i.e., two or more *Ghosts* can simultaneously occupy the same cell of the game grid. Additionally, there are black squares that represent the pellets and dark gray blocks of walls.

*Pac-Man* moves at double the *Ghosts'* speed and since there are no dead ends, it is impossible for a single *Ghost* to complete the task of killing it. Since *Pac-Man* moves faster than a *Ghost*, the only effective way to kill a well-performing *Pac-Man* is for a group of *Ghosts* to hunt cooperatively.

### Pac-Man

Both the difficulty and, to a lesser degree, the interest of the game are directly affected by the intelligence of the *Pac-Man* player. In Yannakakis and Hallam (2004a; 2005), three fixed *Ghost*-avoidance and pellet-eating strategies for the *Pac-Man* player, differing in complexity and effectiveness, are presented. Each strategy is based on decision making applying a cost or probability approximation to the player's four neighboring cells (i.e., up, down, left, and right). We present them briefly in this article.

- Cost-Based (CB) *Pac-Man*: Moves towards a cost minimization path that produces effective *Ghost*-avoidance and (to a lesser degree) pellet-eating behaviors, but only in the local neighbor cell area.
- Rule-Based (RB) *Pac-Man*: This is a CB *Pac-Man*, plus an additional rule for more effective and global pellet-eating behavior.
- Advanced (ADV) *Pac-Man*: The ADV moving strategy generates a more global *Ghost*-avoidance behavior built upon the RB *Pac-Man*'s good pellet-eating strategy.

### **Ghosts**

The arcade version of Pac-Man uses a handful of simple rules and scripted sequences of actions that control each *Ghost* individually (Wikipedia), combined with some random decision making to make the *Ghosts*' behavior less predictable. Even though this design yields quite complex *Ghost* behaviors, the player's interest decreases at the point where *Ghosts* are too fast to beat (Rabin 2002). In our Pac-Man version, we require *Ghosts* to keep learning and constantly adapting to the player's strategy, instead of being opponents with fixed strategies and furthermore maintain a constant real-time speed.

Neural networks are a suitable host for emergent adaptive behaviors in complex multi-agent environments (Ackley and Littman 1992) and have been successfully applied for adaptive learning in real time in computer games (Stanley et al. 2005). A multi-layered fully connected feedforward neural controller, where the sigmoid function is employed at each neuron, manages the *Ghosts*' motion. Using their sensors, *Ghosts* inspect the environment from their own point of view and decide their next action. Each *Ghost*'s perceived input consists of the relative coordinates of *Pac-Man* and the closest *Ghost*. We deliberately exclude from consideration any global sensing, e.g., information about the dispersion of the *Ghosts* as a whole, because we are interested specifically in the minimal sensing scenario. The neural network's output is a four-dimensional vector with respective values from 0 to 1 that represents the *Ghost*'s four movement options (up, down, left, and right, respectively). Each *Ghost* moves towards the available—unobstructed by walls—direction represented by the highest output value. Available movements include the *Ghost*'s previous cell position.

### **Fixed Strategy Ghosts**

Apart from the neural controlled *Ghosts*, three additional non-evolving strategies have been tested for controlling the *Ghost*'s motion. These strategies are used as baseline behaviors for comparison with any neural-controller emerged behavior.

- Random (R): *Ghosts* that randomly decide their next available movement. Available movements have equal probabilities to be picked.
- Followers (F): *Ghosts* designed to follow *Pac-Man* constantly. Their strategy is based on moving so as to reduce the greatest of their relative coordinates from *Pac-Man*.
- Near-Optimal (O): A *Ghost* strategy designed to produce attractive forces between *Ghosts* and *Pac-Man*, as well as repulsive forces among the *Ghosts*.<sup>1</sup>

## ADJUSTING INTEREST PARAMETER VALUES FOR PAC-MAN

In this section, we present the procedures followed to obtain the appropriate parameter values of the interest estimate (5) for the Pac-Man game. In this article  $t_{\min}$  is 35 simulation steps, which is obtained as the minimum simulation time that *Pac-Man* survives when playing against the best-performing near-optimal *Ghosts*. In addition,  $t_{\max}$  is 320 simulation steps, which corresponds to the minimum simulation period required by the RB *Pac-Man* (best pellet-eater) to clear the stage of pellets.

In order to obtain values for the interest formula weighting parameters  $p_1$ ,  $p_2$ , and  $p_3$ , we select empirical values based on each interest criterion. For the first interest metric presented in (1),  $p_1$  is adjusted so as to give  $T$  a greater impact or else a boost when even a slight difference between the maximum and the average lifetime of the player (i.e., challenge) is noted ( $p_1 < 1$ ). This way we reward quite challengeable opponents more than near-optimal killers. For the third interest metric presented in (4),  $p_3$  is adjusted in order to press for very high  $H_n$  values and furthermore to provide a clearer distinction between high and low normalized entropy values ( $p_3 > 1$ ). Finally,  $p_2$  is set so as  $\sigma$  has a linear effect on  $S$ . By taking this into consideration,  $p_1 = 0.5$ ,  $p_2 = 1$ , and  $p_3 = 4$ , for the experiments presented in this article.

Moreover, values for the interest criteria weighting parameters  $\gamma$ ,  $\delta$ , and  $\epsilon$  are also selected empirically based on the specific game. In Pac-Man, aggressive opponent behavior is of the greatest interest. The game loses any reliability when *Ghosts* stick in a corner instead of wandering around the stage. Thus, diversity in gameplay ( $S$ ) and appropriate level of challenge ( $T$ ) should come next in the importance list of interest criteria. Given the previously mentioned statements and by adjusting these three parameters so that the interest value escalates as the opponent behavior changes from Random to near-optimal, and then to follower, we come up with  $\gamma = 1$ ,  $\delta = 2$  and  $\epsilon = 3$ .

Since the interest value changes monotonically with respect to each of the three criterion values  $T$ ,  $S$ ,  $E\{H_n\}$ , sensitivity analysis is conducted on

the interest metric parameters, aiming to portray the relation between these parameters as well as their weighting degree in the interest formula. We therefore proceed by seeking opponent behaviors that generate ten different  $T$ ,  $S$ , and  $E\{H_n\}$  values, equally spread in the  $[0,1]$  interval. Given these thirty values as input,  $p_1$ ,  $p_2$ ,  $p_3$ ,  $\gamma$ ,  $\delta$ , and  $\epsilon$  parameters are systematically changed one at a time so that their percentage difference lies in the interval  $[-50\%, 50\%]$ . Each time a parameter change occurs, the absolute percentage difference of the game's interest is computed. The function between the absolute percentage differences of the interest value and the percentage differences of the interest weighting parameters is illustrated in Figure 2.

As seen in Figure 2, changes on the  $p_1$ ,  $p_2$ , and  $p_3$  parameters seem to influence the  $I$  value more than  $\gamma$ ,  $\delta$ , and  $\epsilon$ . The observed difference in interest sensitivity is reasonable since the first three parameters represent powers, while the latter three correspond to product weights. More specifically,  $p_2$  and  $p_3$  reveal significant differences (i.e., greater than 5%) in  $I$  when decreased by 15% (i.e.,  $p_2 = 0.85$ ) and 9% (i.e.,  $p_3 = 3.64$ ) or increased by 20% (i.e.,  $p_2 = 1.2$ ) and 10% (i.e.,  $p_3 = 4.4$ ), respectively. For  $p_1$  significant change in  $I$  is observed only when decreased by up to 35% (i.e.,  $p_1 = 0.325$ ). Accordingly, both  $\epsilon$  and  $\delta$  parameters reveal significant differences in  $I$  only when decreased by 40% and 45% respectively. Finally, for  $\gamma$  no significant change in  $I$  is observed even when changed by up to 50%.

Regardless of the sensitivity of the  $I$  value, as far as mainly the  $p_2$  and  $p_3$  parameters are concerned, we believe that the selected values project a robust  $I$  value considering that they constitute power parameters in the interest formula.

## MEASURING PERFORMANCE

When a predator/prey game is investigated from the predator's viewpoint, optimality can be measured in the predators' ability to kill the prey. Thus, prey-killing ability of the *Ghosts* is the primary factor that determines how well-performed a behavior is in the *Pac-Man* game. Furthermore, preventing *Pac-Man* from eating pellets, which also implies fast-killing capabilities, constitutes an additional factor of the desired optimal behavior. Given these, a measure designed to give an approximation of a group of *Ghosts*' performance over a specific number  $N$  of games played, is

$$P = \frac{\alpha(k/N) + \beta \min\{1 + (e_{\min} - E\{e\})/e_{\max} - e_{\min}), 1\}}{\alpha + \beta}, \quad (6)$$

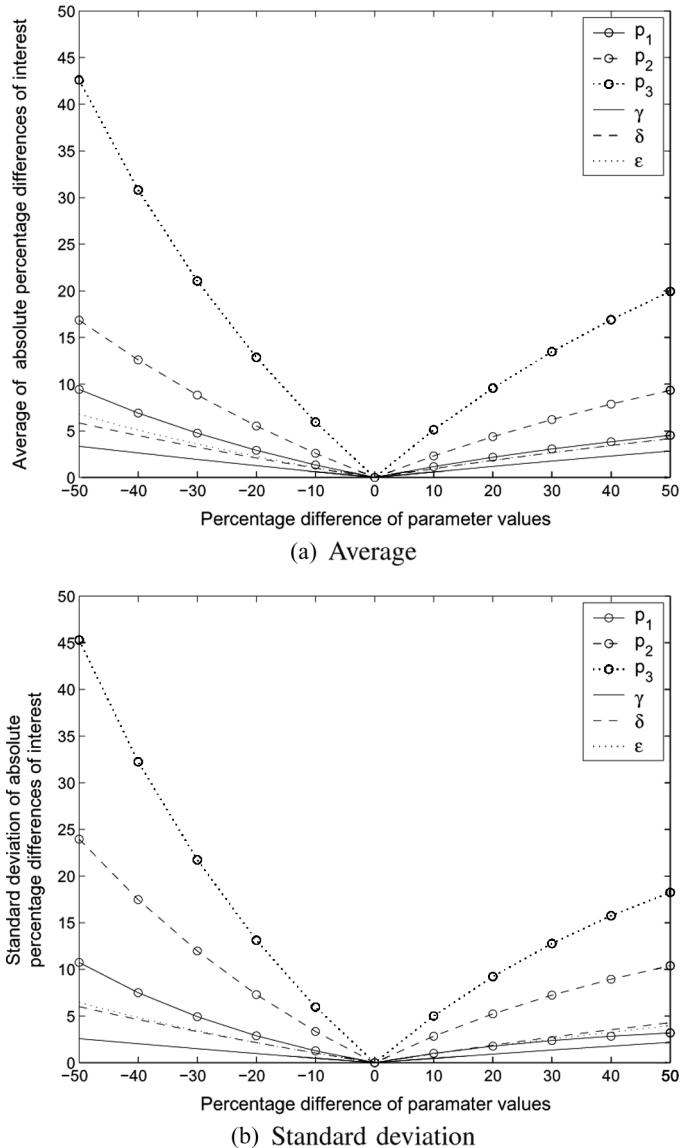


FIGURE 2 Absolute percentage differences of  $I$  over ten runs for each weighting parameter.

where  $P$  is the performance of a *Ghost* group behavior taking values from 0 to 1;  $k$  is the number of *Pac-Man* kills within  $N$  games;  $E\{e\}$  is the average number of pellets eaten by *Pac-Man* over the  $N$  games;  $e_{\min}$ ,  $e_{\max}$  are the lower and upper bound of the eaten pellets  $e$ , respectively (in this article  $e_{\min} = 80$ ,  $e_{\max} = 227$ );  $\alpha$ ,  $\beta$  are weight parameters (in this article  $\alpha = \beta = 1$ ).

## OFFLINE LEARNING

We use an offline evolutionary learning approach in order to produce some “good” (i.e., in terms of performance) initial behaviors. An additional aim of the proposed algorithm is to emerge dissimilar behaviors of high fitness—varying from blocking to aggressive—offering diverse seeds for the online learning mechanism in its attempt to generate emergent *Ghost* behaviors that make the game interesting. The offline learning mechanism used is presented in Yannakakis and Hallam (2004a) and Yannakakis et al. (2004c).

## ONLINE LEARNING (OLL)

As previously noted, games which can learn and adapt to new playing strategies offer a richer interaction to entertain the player. For that purpose, we use an evolutionary machine learning mechanism for the Pac-Man game, which is based on the idea of *Ghosts* that learn while they are playing against *Pac-Man*. Or else, *Ghosts* that are reactive to any player’s behavior learn from its strategy instead of being opponents with fixed strategies that exist in all versions of this game today. Furthermore, this approach’s additional objectives are to keep the game’s interest at high levels as long as it is being played and to achieve good real-time performance (i.e., low computational effort during gameplay). This approach is first introduced in Yannakakis et al. (2004a) for a prey-predator game called “Dead-End” and in (Yannakakis and Hallam (2004a) for the Pac-Man game.

Beginning from any initial group of OLT *Ghosts*, the OLL mechanism transforms them into a group of heterogeneous opponents that are conceptually more interesting to play against. An OLT *Ghost* is cloned four times and its clones are placed in the game field to play against a selected fixed *Pac-Man* type in a selected stage. Then, at each generation:

Step 1. Each *Ghost* is evaluated every  $t$  simulation steps via (7), while the game is played— $t = 50$  simulation steps in this article.

$$f' = \sum_{i=1}^{t/2} \{d_{P,i} - d_{P,(2i-1)}\}, \quad (7)$$

where  $d_{P,i}$  is the distance between the *Ghost* and *Pac-Man* at the  $i$  simulation step. This fitness function promotes *Ghosts* that move towards *Pac-Man* within an evaluation period of  $t$  simulation steps.

Step 2. A pure elitism selection method is used where only the fittest solution is able to breed. The fittest parent clones an offspring with a probability  $p_c$  that is inversely proportional to the normalized cell visit entropy (i.e.,  $p_c = 1 - H_n$ ) given by (4). In other words, the higher the cell visit entropy of the *Ghosts*, the lower the probability of breeding new solutions. If there is no cloning, then go back to Step 1, else continue to Step 3.

Step 3. Mutation occurs in each gene (connection weight) of the offspring's genome with a small probability  $p_m$  (e.g., 0.02). A Gaussian random distribution is used to define the mutated value of the connection weight. The mutated value is obtained from (8).

$$w_m = \mathcal{N}(w, 1 - H_n), \quad (8)$$

where  $w_m$  is the mutated connection weight value and  $w$  is the connection weight value to be mutated. The Gaussian mutation, presented in (8), suggests that the higher the normalized entropy of a group of *Ghosts*, the smaller the variance of the Gaussian distribution and therefore, the less disruptive the mutation process as well as the finer the precision of the GA.

Step 4. The mutated offspring is evaluated briefly via (7) in offline mode, that is, by replacing the least-fit member of the population and playing an offline (i.e., no visualization of the actions) short game of  $t$  simulation steps. If there is a human playing Pac-Man, then the *Pac-Man*'s motion trail of the last  $t$  simulation steps is recorded and opponents are evaluated against it in offline mode. The fitness values of the mutated offspring and the least-fit *Ghost* are compared and the better one is kept for the next generation. This pre-evaluation procedure for the mutated offspring attempts to minimize the probability of group behavior disruption by low-performance mutants. The fact that each mutant's behavior is not tested in a single-agent environment but within a group of heterogeneous *Ghosts*, helps more towards this direction. If the least-fit *Ghost* is replaced, then the mutated offspring takes its position in the game field as well.

The algorithm is terminated when a predetermined number of games has been played or a game of high interest (e.g.,  $I \geq 0.7$ ) is found.

We mainly use short simulation periods ( $t = 50$ ) in order to evaluate *Ghosts* in OLL, aiming to the acceleration of the online evolutionary process. The same period is used for the evaluation of mutated offspring; this is based on two primary objectives: 1) to apply a fair comparison between the mutated offspring and the least-fit *Ghost* (i.e., same evaluation period) and

2) to avoid undesired high computational effort in online mode (i.e., while playing).

## EXPERIMENTS AGAINST FIXED PLAYING STRATEGIES

Results obtained from experiments applied on the *Pac-Man* game against fixed playing strategies are presented in this section. These include offline training and online learning emergent behavior analysis as well as experiments for testing robustness and adaptability of the online learning approach proposed.

### Offline Training

The procedure presented in this subsection is focused on generating well-behaved *Ghosts* in terms of the performance measure described previously. We train *Ghosts* against all three types of *Pac-Man* player through the neuro-evolution offline learning mechanism presented in previously.

In order to minimize the non-deterministic effect of the *Pac-Man*'s strategy on the *Ghost's* performance and interest values as well as to draw a clear picture of these averages' distribution, we apply the following bootstrapping procedure. Using a uniform random distribution we pick 10 different 50-tuples out of the 100 previously mentioned games. These 10 samples of data (i.e.,  $e, k, t_k, v_{ik}$ ) from 50 games (i.e.,  $N = 50$ ) are used to determine the *Ghost's* average performance and interest values and their respective confidence intervals. The outcome of this experiment is presented in Table 1.

**TABLE 1** Performance ( $P$ ) and Interest ( $I$ ) Values (Average Values of 10 Samples of 50 Games Each) of Fixed Strategy (R, F, O) and OLT *Ghosts* (B, A, H) Playing against All Three *Pac-Man* Types (CB, RB, ADV). Average  $P$  and  $I$  Values ( $E\{\cdot\}$ ) of All Six Strategies Appear in the Bottom Row. Experiment Parameters: Population Size is 80,  $g = 1000$ ,  $t = 320$  Simulation Steps.  $N_t$  = Games,  $p_m = 0.02$ , 5-Hidden Neurons Controller

| Trained offline by playing against |       |       |       |       |       |       |
|------------------------------------|-------|-------|-------|-------|-------|-------|
|                                    | CB    |       | RB    |       | ADV   |       |
|                                    | $P$   | $I$   | $P$   | $I$   | $P$   | $I$   |
| R                                  | 0.423 | 0.547 | 0.363 | 0.586 | 0.356 | 0.523 |
| F                                  | 0.754 | 0.771 | 0.701 | 0.772 | 0.621 | 0.771 |
| O                                  | 0.891 | 0.729 | 0.897 | 0.749 | 0.964 | 0.686 |
| B                                  | 0.734 | 0.576 | 0.689 | 0.412 | 0.869 | 0.442 |
| A                                  | 0.661 | 0.654 | 0.606 | 0.652 | 0.662 | 0.555 |
| H                                  | 0.348 | 0.190 | 0.310 | 0.250 | 0.467 | 0.423 |
| $E\{\cdot\}$                       | 0.635 | 0.578 | 0.592 | 0.570 | 0.656 | 0.566 |

Offline trained emergent solutions are the OLL mechanisms' initial points in the search for more interesting games. OLT obtained behaviors are classified into the following categories.

- Blocking (B): These are the OLT *Ghosts* that achieve the best performance against each *Pac-Man* type. Their behavior is characterized as ‘Blocking’ because they tend to wait for *Pac-Man* to enter a specific area that is easy for them to block and then kill. Their average normalized cell visit entropy value  $E\{H_n\}$  lies between 0.55 and 0.65.
- Aggressive (A): These are OLT *Ghosts* that achieve lower performance in comparison to the blockers. Their behavior is characterized as “aggressive” because they tend to follow *Pac-Man* all over the stage in order to kill it. This motion feature generates the highest  $I$  value ( $E\{H_n\} \geq 0.65$ ) among the interest values generated by the three emergent behaviors.
- Hybrid (H): These are suboptimal OLT *Ghosts* that achieve the lowest performance ( $P \leq 0.55$ ) and low interest value in comparison to the aforementioned B and A *Ghosts* ( $E\{H_n\} < 0.55$ ). Their behavior is characterized as “hybrid” because they tend to behave as a blocking-aggressive hybrid, which proves to be ineffective at killing *Pac-Man*.

As far as the interest value generated by the mentioned behaviors is concerned, confidence intervals ( $\pm 0.0647$  maximum,  $\pm 0.0313$  on average) obtained by the bootstrapping procedure previously described indicate that B, A, and H are significantly different.

According to Table 1, near-optimal and blocking behavior *Ghosts* achieve high-performance values against all three *Pac-Man* types, whereas their interest value is not as high as their performance value. This reveals the compromise between optimality and interest it has to be made because, in a predator/prey computer game, optimal killing behaviors are almost never interesting behaviors. On the other hand, followers are likely to produce the most interesting behaviors (among the behaviors examined in Table 1) for the game.

Viewing results presented in Table 1 from the *Pac-Man* type perspective (i.e., the average values in the bottom row of the table), it looks as if the RB and ADV are, respectively, the hardest and easiest *Pac-Man* players to kill. Concerning the three *Pac-Man* types' generated interest, it seems that there is no significant difference amongst them.

## Online Learning Experiments

As previously mentioned, the offline learning procedure is a mechanism that produces near-optimal solutions to the problem of killing

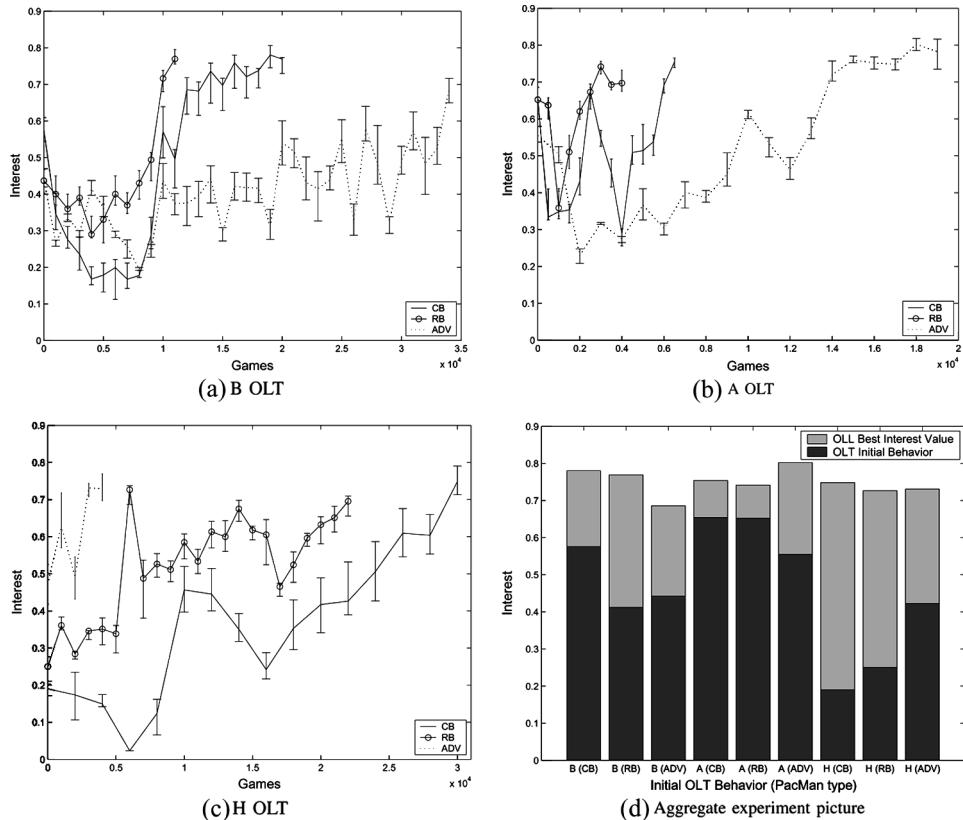
*Pac-Man* and minimizing the pellets eaten in a game. These solutions are the OLL mechanisms' initial points in the search for more interesting games. The OLL experiment is described as follows.

- Pick the nine emerged *Ghosts*' behaviors produced from the offline learning experiments presented previously (i.e., B, A and H behaviors emerged by playing against each *Pac-Man* type).
- Starting from each OLT behavior, apply the OLL mechanism by playing against the same type of *Pac-Man* as was used offline.
- Calculate the interest (bootstrapping procedure with  $N = 50$  of the game every 500 games during each OLL attempt.

The evolution of interest over the OLL games of each one of the three OLT behaviors is presented in a subfigure of Figure 3. For each of the three subfigures, three lines are illustrated, representing the interest values and their respective confidence intervals of the OLL attempt playing against the three *Pac-Man* types. Figure 3(d) illustrates the overall picture of the OLL experiments by comparing the initial interest of the game against the best average interest value achieved from OLL. Clearly, the OLL approach constitutes a robust mechanism that, starting from near-optimal or suboptimal *Ghosts*, manages to emerge interesting games (i.e., interesting *Ghosts*) in the vast majority of cases (i.e., in 8 out of 9 cases  $I > 0.7$ ). In 5 out of 9 OLL attempts, the best interest value is significantly greater or statistically equal to the respective follower's value (i.e., 0.771 against CB, 0.772 against RB and 0.771 against ADV).

As seen from Figure 3, OLL enhances the interest of the game independently of the initial OLT behavior or the *Pac-Man* player *Ghosts* play against. In all experiments presented here, the learning mechanism is capable of producing games of higher than the initial interest as well as keeping that high interest for a long period. There is obviously a slight probability of disruptive mutations (the higher the game's interest, through the cell visit entropy value, the less the probability of mutation) that can cause undesired drops in the game's interest. However, OLL is robust enough to recover from such disruptive phenomena (Figure 3).

Given an interesting initial behavior (e.g., aggressive behavior,  $I > 0.6$ ), it takes some few thousands of games for the learning mechanism to produce games of high interest. On the other hand, it takes some several thousand games to transform an uninteresting near-optimal blocking behavior (see Figure 3(a) and Figure 3(c)) into an interesting one. That is because the OLL process requires an initial long period to disrupt the features of an uninteresting blocking behavior, in order to be able to boost the interest of the game.



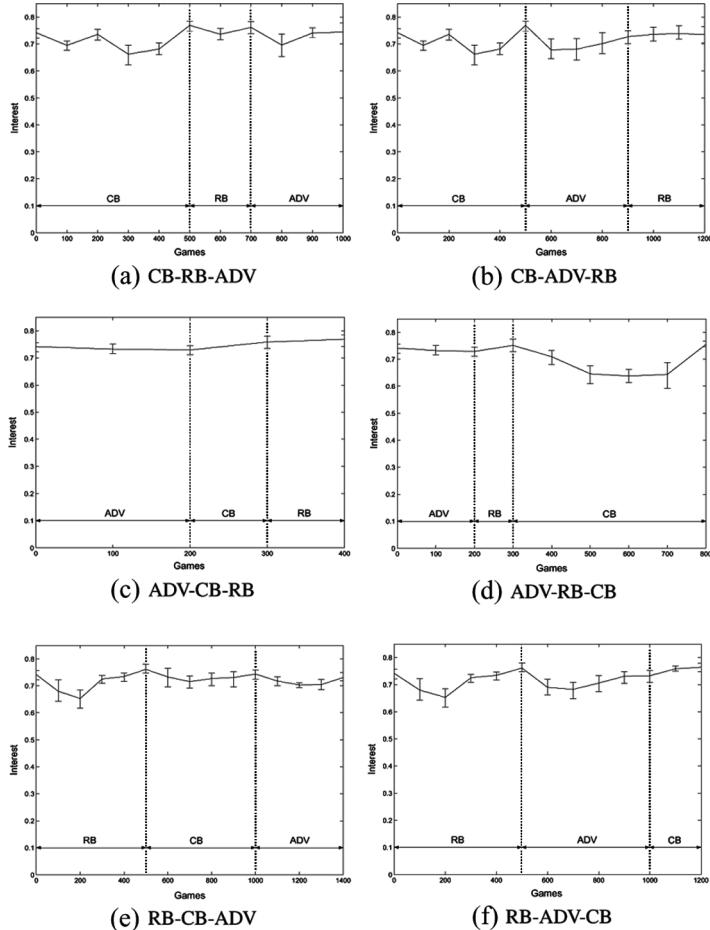
**FIGURE 3** Game interest over the number of OLL games. For reasons of computational effort, the OLL procedure continues for a number of games large enough to illustrate its behavior, after a game of high interest ( $I \geq 0.7$ ) is found. Initial *Ghost* behaviors appear in (a), (b), and (c) subfigure caption, whereas (d) illustrates the overall picture of the experiment. Experiment parameters:  $t = 50$  simulation steps,  $p_m = 0.02$ , 5-hidden neurons controller.

It is obvious that a number in the scale of  $10^3$  constitutes an unrealistic number of games for a human player to play. On that basis, it is very unlikely for a human to play so many games in order to notice the game's interest increasing. The reason for the OLL process being that slow is a matter of keeping the right balance between the process' speed and its "smoothness" (by smoothness we define the interest's magnitude of change over the games). A solution to this problem is to consider the initial long period of disruption as an offline learning procedure and start playing as soon as the game's interest is increased. Moreover, other online learning approaches like co-evolution (Demasi and Cruz 2002), dynamic scripting (Spronck et al. 2004), and reinforcement learning (Graepel et al. 2004; Andrade et al. 2005) could, in part, provide a solution for the long convergence times observed.

How effective will this mechanism be in a potential change from a fixed strategy to a dynamic human *Pac-Man* player? The next subsection provides evidence in order to support the answer.

## Adaptability

In order to test the OLL approach's ability to adapt to a changing environment (i.e., change of *Pac-Man* strategy), the following experiment is proposed. Beginning from an initial behavior of high interest value  $I_{init}$ , we apply the OLL mechanism against a specific *Pac-Man* type. During the online process, we keep changing the type of player as soon as interesting



**FIGURE 4** Online learning *Ghosts* playing against changing types of *Pac-Man*. Sub-figure captions indicate the playing *Pac-Man* sequence.

games (i.e.,  $I \geq I_{init}$ ) are produced. The process stops when all three types of players have played the game.

Since we have three types of players, the total number of such experiments is six (all different player type sequences). These experiments illustrate the overall picture of the approach's behavior against any sequence of *Pac-Man* types. As seen in Figure 4, OLL is able to quickly recover a sudden change in the player's strategy and boost the game's interest at high levels after sufficient games have been played (i.e., 100 to 500 games). The mechanism demonstrates a similar adaptive behavior for all six sequences of *Pac-Man* players, which illustrates its independence of the sequence of the changing *Pac-Man* type.

Results obtained from this experiment provide evidence for the approach's ability to adapt to new types of players as well as its efficiency in producing interesting games against humans with dynamic playing strategies.

## EXPERIMENTS AGAINST HUMAN PLAYERS

Experiments against fixed playing strategies portrayed the OLL mechanism's ability to generate interesting *Pac-Man* games. Apart from being fairly robust, the proposed mechanism demonstrated high and fast adaptability to changing types of player (i.e., playing strategies). The next obvious step to take is to let humans judge whether generated games are realistically interesting, or not, and whether OLL indeed enhances the level of entertainment during play. For this, we conducted a survey, with human subjects as *Pac-Man* players, that primarily aimed to obtain answers for the following questions.

1. Does the interest value computed for a game correlate with human judgment of interest?
2. Does the online learning mechanism cause perceived interest to change? Do perceived changes match computed ones?

The experiment is described next. Then the statistical method used and the analysis of obtained results are presented, respectively.

## EXPERIMENT DESCRIPTION

Answers to the target questions presented previously are based on statistical analysis of data acquired from a questionnaire applied for the *Pac-Man* game. The main prerequisite for a subject to participate in this experiment is to have played the original version (Namco) of the *Pac-Man* game at least once. Subject age covers a range between 17 and 51

years, where both sexes are almost equally represented (43.3% females, 56.7% males). In addition, all subjects speak English as a foreign language since their nationality is either Danish (90%) or Greek (10%). The questionnaire is divided into three parts (A, B, and C) and the steps that the subjects go through at each part are presented as follows.

## Personal Data

Subjects are asked to define their interest in the Pac-Man game before they play it. Answers categorize participants into three types of Pac-Man player' represented as "Like," "Neutral," and "Don't like."

Subjects familiarize themselves with the game by playing 50 games against specific OLT opponents (i.e., opponent 4 presented in Table 2). Online learning is used during this testing period, which is not noticeable to the player. At the end of the testing period, each subject's opponents trained online are saved.

## 1st Objective

We pick opponents differing in interest measured against the ADV player (as the most advanced computer-guided *Pac-Man* player). We select five opponents whose interest values uniformly cover the [0, 1] space. The selected opponents' number, which is used as an id-code, and their respective interest values are presented in Table 2.

By experimental design, each subject plays against three of the selected opponents in all permutations of pairs. In addition, we require equal participation of all three player types. For this experiment, we use 30 subjects divided into three equal subsets for each of the three player types (Like, Neutral, Don't Like), since  $C_3^5 = 10$  subjects are required for each player type. Moreover, observed effects show that 30 subjects constitute a statistically significant sample.

**TABLE 2** The Selected Opponents and Their Respective Interest— $I$  and 95% Confidence Interval ( $I_u$ ,  $I_l$ ) Values

| Opponent | Interest |        |        |
|----------|----------|--------|--------|
|          | $I_u$    | $I$    | $I_l$  |
| 1        | 0.2043   | 0.1793 | 0.1494 |
| 2        | 0.3673   | 0.3158 | 0.2670 |
| 3        | 0.5501   | 0.4943 | 0.4420 |
| 4        | 0.6706   | 0.6484 | 0.6267 |
| 5        | 0.8180   | 0.8023 | 0.7858 |

- As previously mentioned, each subject plays sets of games (five games in each set) against three of the selected opponents in all permutations of pairs and each time a pair of sets is completed, the player is asked whether the first set was more interesting than the second set of games. We use the 2-alternative forced choice (2-AFC) approach since it offers several advantages for a subjective interest capture. The 2-AFC comparative fun analysis (Read et al. 2002) minimizes the assumptions about people's different notions of entertainment and provides data for a fair comparison among answers of different people (Yannakakis et al. 2006)

The total number of sets of games that is played by each subject is 12 (all permutations of three pairs. Given thirty subjects, there are nine observed incidents for each pair of sets.

## **2nd Objective**

Each subject plays 25 games against the initial training phase opponents (i.e., opponent 4—OP4) and 25 games against the online trained opponents that were saved (i.e., two sets of games). We let each subject play another two sets against these opponents in different order. Half of the subjects play these four sets of games in the sequence OLL-OP4, OP4-OLL, whereas the other half play them in the sequence OP4-OLL, OLL-OP4, since we require minimization of any potential ordering effect. Each time a pair of sets (two pairs here) is finished, the player is asked whether the first set was more interesting than the second set of games.

Subjects are asked to list the criteria they used for their assessment of which set of games was more interesting.

## **STATISTICAL METHOD**

For this experiment, there are three null hypotheses formed.

- $H_0$ : The correlation between observed human judgment of interest and the computed interest value, as far as the different opponents are concerned, is a result of randomness.
- $H_1$ : Observed human judgment of interest does not correlate with the computed interest value, as far as the different opponents are concerned.
- $H_2$ : Observed human judgment of interest does not correlate with performance during play.

Given the interest metric (5) and two sets of games  $A$  and  $B$ , it can be determined that “game A is more (or less) interesting than game B.” In answer to the same question, a human subject can indicate that either

$I_A > I_B$  or  $I_A < I_B$ . In order to measure the degree of agreement between the human judgment of interest and the interest value given by (5), we calculate the correlation coefficients

$$c(\vec{z}) = \sum_{i=1}^N z_i / N, \quad (9)$$

where  $N$  is the number of incidents to correlate and

$$\vec{z} = \begin{cases} 1, & \text{if subject agrees with (5);} \\ -1, & \text{if subject disagrees with (5).} \end{cases} \quad (10)$$

The test statistic (9) is used to assess the truth of all three null hypotheses. However, for the null hypothesis  $H_2$ , the correlation coefficients  $c(\vec{z}')$  are computed where  $z'$  values are obtained from (11).

$$\vec{z}' = \begin{cases} 1, & \text{if subject chooses according to performance;} \\ -1, & \text{if subject does not choose according to performance.} \end{cases} \quad (11)$$

The distribution used for obtaining the correlation coefficient probabilities (p-values— $P(C \geq c)$ ) is the binomial. The observed effect is “highly significant” and “significant” if  $P(C \geq c) < 1\%$  and  $1\% < P(C \geq c) \leq 5\%$ , respectively.

For the design of the subjects’s self reports we follow the principles of comparative fun analysis (Read et al. 2002; Yannakakis et al. 2006). The endurability and expectations for the majority of subjects that played Pac-Man were very high, indicating that the game design used was successful. More specifically, all subjects were excited to play Pac-Man as soon as they were informed about the rules of the game (derived through a *Funometer* tool application [Read et al. 2002]) and the majority of subjects stressed that they would like to play the game again (derived through an Again-Again table [Read et al. 2002]). As previously mentioned, we use the 2-alternative forced choice (2-AFC) approach since it offers several advantages for a subjective entertainment capture. The 2-AFC comparative fun analysis minimizes the assumptions about people’s different notions of entertainment and provides data for a fair comparison among answers of different people.

## STATISTICAL ANALYSIS

As noted previously, this article concentrates on the characters’ behavioral aspect of interesting games. More specifically, it focuses on the opponent’s rather than the graphics’ or the sound’s impact on the player’s entertainment. Apart from the opponent, there are two additional factors that may affect the interest of a computer game, that are examined in this

section. These are the player-subject type (degree of *a priori* game liking) and the order of play.

## Opponent

Each entity in Table 3 represents a subject's answer to the question from the 1st objective, equivalent to "Is  $I_i > I_j?$ ," where  $i, j$ , are the row and column number, respectively. Given the interest values of the five opponents (see Table 2), "O and X" stand, respectively, for the subject's agreement and disagreement with this ranking (in other words, O and X characters are selected for visual purposes to symbolize the respective  $z$  values—see (10). As stressed before, given 30 subjects, there are nine incidents for each pair of opponent which are represented in a  $3 \times 3$  matrix. Rows within this matrix denote the type of subject that answered the specific question.

Table 4 presents the correlation coefficients and their respective  $P(C \geq c)$  values for each one of the ten combinations of opponent pairs ( $N = 18$ ) and in total ( $N = 180$ ). There is an obvious disagreement between the interest metric and the human's notion of interest in opponent pairs 1–2 and 3–4. Even though humans seem to agree with the interest metric in the pairs 1–3 and 1–4, the obtained p-values reveal statistically insignificant results. For the rest of the pairs, we experience statistically highly significant (i.e., 2–3, 2–5, 4–5) and significant (i.e., 1–5, 2–4, 3–5) matching to

**TABLE 3** Agreement Between the Subject's Judgment of Interest and the Interest Metric—O:  $z = 1$ , X :  $z = -1$

|              |            | Is $I_{Row} > I_{Column}?$ |       |       |       |       |
|--------------|------------|----------------------------|-------|-------|-------|-------|
|              |            | 1                          | 2     | 3     | 4     | 5     |
| Subject type |            |                            |       |       |       |       |
| 1            | Like       |                            | O O X | O O X | O O X | O O O |
|              | Neutral    |                            | O O X | O O O | O O X | O O X |
|              | Don't Like |                            | O O O | O O O | O O O | O X X |
| 2            | Like       | X X X                      |       | O O O | O O X | O O O |
|              | Neutral    | X X X                      |       | O O X | O O O | O O X |
|              | Don't Like | O X X                      |       | O O O | O O X | O O X |
| 3            | Like       | O X X                      | O O O |       | O X X | O O X |
|              | Neutral    | O O X                      | O O X |       | O X X | O O O |
|              | Don't Like | O X X                      | O O X |       | O O X | O O X |
| 4            | Like       | O O X                      | O O O | X X X |       | O O X |
|              | Neutral    | O O X                      | O O X | X X X |       | O O X |
|              | Don't Like | O X X                      | O O X | O X X |       | O O O |
| 5            | Like       | O O X                      | O O O | O O O | O O O |       |
|              | Neutral    | O O O                      | O O X | O O X | O O O |       |
|              | Don't Like | O O O                      | O O O | O O X | O O O |       |

**TABLE 4** Interest Metric – Subject Judgment Correlation Coefficients  $c$ ,  $P(C \geq c)$  Values, Order of Play Test Statistic  $z''$ , and  $P(Z \geq |z''|)$  Values for all Pairs of Opponents and in Total

| Pair  | $c$    | $P(C \geq c)$        | $z''$  | $P(Z \geq  z'' )$ |
|-------|--------|----------------------|--------|-------------------|
| 1–2   | −0.111 | 0.7596               | 0.2222 | 0.2403            |
| 1–3   | 0.3333 | 0.1189               | 0.3333 | 0.1189            |
| 1–4   | 0.3333 | 0.1189               | −0.222 | 0.2403            |
| 1–5   | 0.5555 | 0.0154               | −0.111 | 0.4072            |
| 2–3   | 0.6666 | 0.0037               | 0.1111 | 0.4072            |
| 2–4   | 0.5555 | 0.0154               | 0.1111 | 0.4072            |
| 2–5   | 0.7777 | 0.0006               | 0.0000 | 0.5927            |
| 3–4   | −0.444 | 0.9845               | −0.111 | 0.4072            |
| 3–5   | 0.5555 | 0.0154               | −0.333 | 0.1189            |
| 4–5   | 0.6666 | 0.0037               | 0.2222 | 0.2403            |
| Total | 0.3888 | $1.31 \cdot 10^{-7}$ | 0.0222 | 0.4818            |

observed human judgment. Finally, the total agreement correlation coefficient ( $c = 0.3888$ ), as well as its p-value ( $P(C \geq c) = 1.31 \cdot 10^{-7}$ ), demonstrate a statistically highly significant effect that rules out the null hypothesis  $H_1$ . Thus, it appears that the observed human judgment of interest correlates with the computed interest value, as far as the different opponents are concerned. Moreover, the obtained p-values presented in Table 4 illustrate that the sample size of 30 subjects is adequate to produce statistically significant observed effects.

### ***Opponent 1***

Further investigation of the interest value generated by opponent 1 showed high dependence on the player type. More specifically, when opponent 1 plays against the CB *Pac-Man* and the RB *Pac-Man*, it generates interest, which is respectively statistically not different and significantly higher than the interest generated by opponent 2. Opponent 1 constitutes a particular case since no such change in the opponent ranking (i.e., ranked by interest) occurs for any other of the four remaining opponents.

Given the ranking instability of opponent 1, we recalculate the  $z$  values as if (1)  $I_1 > I_2$  and (2)  $I_1 = I_2$  and proceed. In the former case, the  $z$  values of the [1–2] pair swap their sign and the obtained p-values for this pair and in total are 0.4072 and  $2.57 \cdot 10^{-8}$ , respectively. For the latter case, the  $z$  values of the [1–2] pair are not taken into consideration and the two first (triplets of) rows and columns of Table 3 are merged into one by adding up their  $z$  values. The obtained p-values for the merged pairs and in total are presented in Table 5. For both cases, changes in the opponent 1 ranking increase the significance of the observed effects.

**TABLE 5** Interest Metric: Subject Judgment Correlation Coefficients  $c$  and  $P(C \geq c)$  Values When  $I_1 = I_2$  Is Assumed

| Pairs   | $c$    | $P(C \geq c)$        |
|---------|--------|----------------------|
| (1,2)-3 | 0.5000 | 0.0019               |
| (1,2)-4 | 0.4444 | 0.0056               |
| (1,2)-5 | 0.6667 | $3.5 \cdot 10^{-5}$  |
| Total   | 0.4444 | $1.17 \cdot 10^{-8}$ |

## Order of Play

In order to check whether the order of playing Pac-Man games affects the human judgment of interest, we hypothesize that there is no order effect and proceed as follows. For each pair of opponents, that a subject played in both orders, we count (a) the times  $K$  that the subject agrees with the interest value only in the first pair played and (b) the times  $J$  that the subject agrees with the interest value only in the latter pair played. In the case where the subject agrees or disagrees with the interest value in both pairs played, we take no action. To this end, we compute the  $z''$  value (12) for each pair of opponents ( $N = 9$ ) and in total ( $N = 90$ ).

$$z''(K, J) = (K - J)/N \quad (12)$$

The greater the absolute value of  $z''(K, J)$ , the more the order of play tends to affect the subjects' judgment of interest. This value defines the test statistic used to assess the truth of the hypothesis that there is no order effect. The obtained  $z''$  value is trinomially distributed.

As seen from Table 4, there are no statistically significant effects in any pair of opponents or in total. Therefore, the null hypothesis is not rejected and it seems that the order of play does not affect the human judgement of interest.

## Opponent 1

The order of play is not affected by the particular behavior of opponent 1 either. That is, if  $I_1 > I_2$ , there is no difference in the obtained  $P(Z \geq |z''|)$  values and, if  $I_1 = I_2$ , there is no statistically significant effects in any pair of opponents or in total (i.e.,  $P(Z \geq |z''|) = 0.5312$ ).

## Subject Type

In this section, we present how the subject's type, which corresponds to the subject's "liking of the Pac-Man game," correlates with the subject's judgment of interest. To this end, we compute the correlation coefficients  $c$  and their respective probabilities  $P(C \geq c)$  for each subject type (60 incidents for each type).

**TABLE 6** Interest Metric – Subject Judgment Correlation  
 Coefficients  $c$ ,  $P(C \geq c)$  Values of the Three Types of Subject and  
 Correlation Variance ( $\sigma_c^2$ ) Over the 10 Subjects of Each Type

| Subject type | $c$    | $(\sigma_c^2)$ | $P(C \geq c)$        |
|--------------|--------|----------------|----------------------|
| Like         | 0.4000 | 0.0691         | 0.0013               |
| Neutral      | 0.6333 | 0.1234         | 0.0067               |
| Don't like   | 0.4333 | 0.1493         | 0.0005               |
| Total        | 0.3888 | 0.1079         | $1.31 \cdot 10^{-7}$ |

As seen from Table 6, all three types of subject's observed judgment of interest collectively demonstrate a highly significant agreement ( $P < 1\%$ ) with the interest metric. However, it appears that there is no significant difference between the three types and, therefore, no secure conclusions about the subject's type effect on its notion of interest can be arisen.

### *Opponent 1*

By following the procedure described previously, for the particular case of opponent 1, we also come up with highly significant values for all three subject types and no significant difference between them for both cases of  $I_1 > I_2$  and  $I_1 = I_2$ .

### **Online Learning**

In this section, we analyze the observed effects from the on-line learning experiment (Part C) presented previously. In Part C, subjects play 2 sets of 50 games in total. Interest values calculated (bootstrapping procedure with  $N = 25$ ) and presented in Table 7 show that, in 18 out of 30 cases, the human player managed to produce more interesting games by the use of the online learning procedure. However, it is not clear whether OLL used in humans cause the interest value to proliferate. Thus, it seems that 50 OLL games (testing period in Part A) are not adequate for the OLL mechanism to cause a significant difference in the interest value (see Table 7).

Choosing an online learning period (or else testing period) of 50 games is an empirical way of balancing efficiency and experimental time. The duration of the testing period lasted 20 minutes on average, whereas the whole experiment exceeded 65 minutes in many cases, which is a great amount of time for a human to be constantly concentrated. Fixed strategy *Pac-Man* player results showed that more online learning games are required for the interest value to change significantly, which appears to be the case for human players as well.

By calculating the correlation coefficient (9) between the computed interest values (presented in Table 7) and the human judgment of interest

**TABLE 7** Interest  $I$  and Confidence Interval ( $I_u, I_l$ ) Values Against All 30 Human Players Ranked by Subject Type. i.e. 1–10: Like, 11–20: Neutral, 21–30: Don't Like

| Subject      | OLL   |       |       | No OLL |       |       |
|--------------|-------|-------|-------|--------|-------|-------|
|              | $I_u$ | $I_l$ | $I$   | $I_u$  | $I_l$ | $I$   |
| 1            | 0.721 | 0.575 | 0.671 | 0.745  | 0.393 | 0.630 |
| 2            | 0.753 | 0.588 | 0.669 | 0.767  | 0.593 | 0.703 |
| 3            | 0.733 | 0.614 | 0.669 | 0.755  | 0.607 | 0.694 |
| 4            | 0.805 | 0.672 | 0.735 | 0.792  | 0.520 | 0.677 |
| 5            | 0.802 | 0.644 | 0.711 | 0.720  | 0.582 | 0.665 |
| 6            | 0.763 | 0.598 | 0.676 | 0.733  | 0.531 | 0.647 |
| 7            | 0.725 | 0.638 | 0.689 | 0.698  | 0.559 | 0.644 |
| 8            | 0.751 | 0.566 | 0.673 | 0.804  | 0.603 | 0.720 |
| 9            | 0.746 | 0.568 | 0.681 | 0.751  | 0.630 | 0.698 |
| 10           | 0.780 | 0.531 | 0.670 | 0.780  | 0.616 | 0.715 |
| 11           | 0.692 | 0.469 | 0.619 | 0.750  | 0.576 | 0.695 |
| 12           | 0.802 | 0.678 | 0.748 | 0.865  | 0.700 | 0.778 |
| 13           | 0.806 | 0.530 | 0.662 | 0.716  | 0.532 | 0.638 |
| 14           | 0.799 | 0.589 | 0.715 | 0.805  | 0.678 | 0.738 |
| 15           | 0.776 | 0.636 | 0.707 | 0.782  | 0.656 | 0.706 |
| 16           | 0.812 | 0.658 | 0.749 | 0.806  | 0.689 | 0.745 |
| 17           | 0.784 | 0.601 | 0.706 | 0.743  | 0.609 | 0.679 |
| 18           | 0.796 | 0.595 | 0.708 | 0.740  | 0.567 | 0.655 |
| 19           | 0.780 | 0.612 | 0.702 | 0.718  | 0.626 | 0.670 |
| 20           | 0.749 | 0.666 | 0.717 | 0.759  | 0.646 | 0.716 |
| 21           | 0.753 | 0.625 | 0.684 | 0.757  | 0.659 | 0.706 |
| 22           | 0.790 | 0.660 | 0.728 | 0.831  | 0.625 | 0.733 |
| 23           | 0.774 | 0.640 | 0.709 | 0.762  | 0.663 | 0.700 |
| 24           | 0.752 | 0.599 | 0.668 | 0.754  | 0.612 | 0.681 |
| 25           | 0.741 | 0.635 | 0.696 | 0.705  | 0.589 | 0.660 |
| 26           | 0.825 | 0.697 | 0.770 | 0.781  | 0.681 | 0.728 |
| 27           | 0.799 | 0.622 | 0.732 | 0.782  | 0.640 | 0.724 |
| 28           | 0.786 | 0.630 | 0.719 | 0.755  | 0.570 | 0.693 |
| 29           | 0.745 | 0.607 | 0.690 | 0.748  | 0.606 | 0.705 |
| 30           | 0.793 | 0.673 | 0.738 | 0.782  | 0.591 | 0.678 |
| $E\{\cdot\}$ | 0.771 | 0.614 | 0.700 | 0.763  | 0.605 | 0.694 |

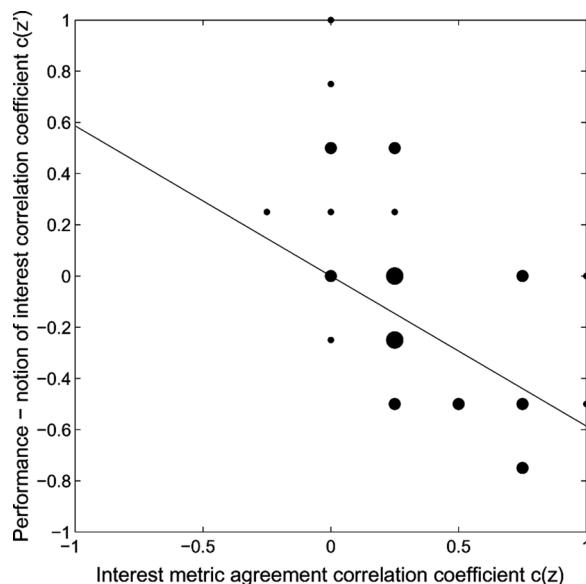
obtained the first question of our second objective, we get a value of  $c = 0.1666$ , with corresponding probability of  $P(C \geq c) = 0.1225$  for  $N = 60$ . This does not constitute a statistically significant effect and suggests that humans were not able to tell the difference between opponent 4 and the opponents trained online at the end of the testing period.

### Performance Factor

As noted before, each subject plays eight pairs of sets of games in total during this experiment (six in Part B and two in Part C), and each set is assigned a score that corresponds to the performance of the subject. More

specifically, the score is directly proportional to the number of pellets eaten by the player. Given the subjects' scores and the reported interest judgement, the  $z'$  values are computed as follows. If the subject chooses the set of games with the higher score obtained as being more interesting, then the  $z'$  value is 1. Accordingly, the  $z'$  value is -1 if the subject chooses the set of games with the lower score obtained as being more interesting. By computing (9) for all 30 subjects ( $N = 8 \cdot 30 = 240$ ), we get  $c(z') = -0.05$  and  $P(C \geq -0.05) = 0.7994$ , which constitutes the effect as statistically not significant. Therefore, the null hypothesis  $H_2$  is not rejected and it seems that observed human judgment of interest does not correlate with performance during play.

However, before abandoning the hypothesis of the performance impact on human judgment totally, we attempt to draw the relation between the two from another perspective. Figure 5 illustrates a scatter plot of the correlation coefficients between the performance and the subject's judgement of interest against the correlation coefficients between the interest metric and the subject's judgment of interest for each subject. In addition, the line of the statistical correlation ( $f(x) = \text{cor}(c(\vec{z}), c(z')) \cdot x$ , where  $\text{cor}(c(\vec{z}), c(z')) = \text{cov}(c(\vec{z}), c(z'))/\sigma_{c(\vec{z})}\sigma_{c(z')} = -0.5864$ ) between the two samples of data is plotted. If we examine Figure 5 in detail as well as the reported interest criteria in the last question of the survey, there seems to be a classification of the subjects into three groups. These are:



**FIGURE 5** Scatter plot of  $c(z)$  and  $c(z')$  values for each subject and their statistical correlation' line. The circular marker' radius is increased in respect to the number of occurrences (i.e., 1, 2 or 3).

- Subjects that judge interest according to their performance ( $c(z') \geq 0.5$ ), size: 6 out of 30 subjects. As far as their agreement with the interest metric is concerned, their observed judgment portrays a rather random behavior ( $0.0 \leq c(z) \leq 0.25$ ). The reported interest criteria in the last question of the survey are explicit. Randomness and scoring performance are the major criteria in selecting the most interesting set between two.
- Subjects that do not judge interest according to their performance ( $c(z') \leq 0.0$ ) and whose interest judgment correlates with the interest metric ( $c(z) \geq 0.5$ ), size: 12 out of 30 subjects. This group's reported interest criteria are focused on the opponent's contribution to the player's satisfaction.
- Subjects that do not judge interest according to their performance ( $-0.5 < c(z') < 0.5$ ) and whose interest judgment does not correlate with the interest metric ( $c(z) < 0.5$ ), size: 12 out of 30 subjects. Subjects of this group concentrate on a variety of Pac-Man aspects different or implicitly syngeneic to the *Ghosts'* behavior, as acquired from the reported interest criteria. These aspects include performance, game control ability, graphics, difficulty, and duration of game.

The computed statistical correlation value and Figure 5 provide evidence that human judgment of interest, that agrees with the interest metric, is not correlated with the human judgment of interest based on performance. In other words, it seems that subjects agreeing with the interest metric do not judge interest by their performance. Or else, subjects disagreeing with the interest metric seem to judge interest by their score and/or other criteria such as game controls and graphics.

### ***Opponent 1***

By assuming that  $I_1 > I_2^2$ , we reveal a slightly higher statistical correlation value  $\text{cor}(c(\vec{z}), c(z')) = -0.5341$ , but conceptually the same effects and subject classification groups as the above-mentioned.

## **CONCLUSIONS**

Predator strategies in predator/prey computer games are still nowadays based on simple rules, which make the game pretty predictable and, therefore, somewhat uninteresting (by the time the player gains more experience and playing skills) (Woodcock 2001; Rabin 2002). A computer game becomes enjoyable primarily when there is a richer online interaction between the player and its opponents who demonstrate interesting behaviors (Yannakakis and Hallam 2004a; 2005c). Machine learning techniques applied online (Stanley et al. 2005) can generate behaviors that give the

illusion of intelligence, which is an important criterion for the human player's perceived entertainment. On top of that, playing against cooperative opponents makes the game more realistic and appealing to the human eye (Yannakakis and Hallam 2005b).

Given some objective criteria for defining interest in predator/prey games, we introduced a generic method for measuring interest in such games. The  $I$  metric presented in this article captures the concept of interest objectively and it is dependent on the player-game opponent interaction. We saw that by using the proposed online learning mechanism on the Pac-Man game, maximization of the individual simple distance measure (see (7)) coincides with maximization of the game's interest. Apart from being fairly robust, the proposed approach demonstrated high and fast adaptability to changing types of player (i.e., playing strategies). Results obtained against fixed strategy *Pac-Man* players showed that such a mechanism could be able to produce interesting interactive opponents (i.e., games) against dynamic human playing strategies.

By testing the game against humans, we managed to confirm our hypothesis that the interest value computed by (5) is consistent with the judgment of human players. In fact, human player's notion of interest of the Pac-Man game correlate highly with the captured interest value. However, there are instances where humans' reported notion of interest does not match the respective calculated  $I$  value of the game. Since the proposed interest metric was designed and evaluated on computer-controlled *Pac-Man* players, the reported mismatches confirm the fact that a human playing behavior differs from a computer-controlled designed player. In addition, it is revealed that both the subject type (i.e., experience/likeness with the game) and the order of playing the game do not affect their judgment. Moreover, given each subject's score, it was demonstrated that humans agreeing with the interest metric do not judge interest by their performance. Or else, humans disagreeing with the interest metric judge interest by their score or based on other personal criteria like game control and graphics.

The main assumption drawn for the interest metric proposed is that players overall have a basic level of gaming skills for the test-bed game. In that sense, the computer-guided players used are models of some well-behaved, average skill players based on similar motion patterns that do not leave much space for significant differences in their best generated interest values. Human players that tested Pac-Man cross-validate this assumption since their generated interest values against the same opponent were not significantly different from each other.

As far as online learning against human players is concerned, results show that more online learning games are required for the interest value to change significantly and for humans to notice some sort of change in

the interest of the game. This is a function of the way that human-game interaction is used to train the opponents. Given more computing power, it may be possible to use the data provided by human-game interaction more efficiently and therefore achieve significant change in interest in fewer games. For instance, thousands of mutants could be evaluated in parallel over longer periods—which would provide better behavior estimates—and moreover the frequency of evolutionary iterations could be increased. Using this approach, we could accelerate the learning where appropriate and minimize the probability of unwanted, unrealistic, non-intelligent generated behaviors due to mutation. Clearly, a single unrealistic emerged AI behavior is sufficient to impair the “intelligent” image any adaptive approach is attempting to present and furthermore to diminish the satisfaction of the player (Champandard 2004; Funge 2004).

## Discussion

As a novel direction in AI in computer games, cooperative opponents that learn in real time for optimizing the entertainment value of the game constitutes this work’s proposed step for future game development. Showing that such a learning mechanism maintains high levels of player satisfaction makes this approach appealing for application to the vast majority of game genres where online learning and opponent cooperation are, until nowadays, deliberately absent or optional. Here we discuss the potential of the methodology in other genres of multi-opponent games where the complication of the opponents’ tasks may differ. More specifically, we analyze the extensibility of the interest metric proposed, the online evolutionary learning mechanism and the neuro-controller used.

1. *Interest Metric* As already mentioned, the criteria of challenge and behavior’s diversity may be effectively applied for measuring the real-time entertainment value of any genre of games. Spatial diversity may in a sense also contribute to the interest value of specific genres (e.g., team-sport, real-time strategy, and first-person shooter games). As long as game developers can determine and extract the features (e.g., through online questionnaires) of the opponent behavior that generate excitement for the player, a mathematical formula can be designed in order to collectively represent them.
2. *Learning Methodology* The proposed online evolutionary learning method may also be successfully applied to any game during active real-time player-opponent interactions. Extracted features of this interaction may be used in order to estimate the fitness of the involved opponents according to their tasks. The replacement of the worst-fit opponent(s) method may be applied in frequent game periods to enhance the

group's fitness. See also Stanley et al. (2005) for a successful application of this method in the NERO game.

Artificial evolution can explore complex search spaces efficiently and, when combined with NNs, it can demonstrate fast adaptability to dynamic and changing environments. Therefore neuro-evolution is recommended for learning in real time. However, convergence time and unpredictability of the emergent behaviors constitute the disadvantages of the methodology, which can be dealt with by careful design of the learning mechanism. The tradeoff between opponent behavior stability and speed of learning online when using neuro-evolution was addressed in this article. Both the breeding of offspring and the variance of the Gaussian mutation used are inversely proportional to the cell visit entropy. This GA scheme minimizes unpredictability and allows for rapid genotype alterations when the interest value of the game is high and low, respectively.

In addition to a careful GA design, player modeling techniques are able to decrease convergence time down to realistic periods of time (i.e., tens of games) and furthermore proliferate the efficiency and justifiability of learning in real time (Yannakakis and Maragondiakis 2005).

3. *Controller* Artificial neural networks serve successfully the adaptability requirements for predator/prey reactive games in real-time. However, as the complexity of the opponents' tasks increases, there might be a need for more sophisticated structures of distributed representation. Memory of previous behaviors learned through the player-opponent interaction may very well be essential when a combination of various tasks is required. Recurrent NNs or augmented NN topologies with hidden states (Stanley and Miikulainen 2002) may be more appropriate when the opponents' tasks proliferate. Moreover, a hierarchy design of neuro-controllers that serve different opponent tasks could also provide the online learning mechanism with more flexibility and faster adaptability. Decision trees, adaptive scripts (Spronck et al. 2001), or classifier systems (Champandard 2004) could also host adaptive behaviors in real time successfully.

## REFERENCES

- Ackley, D. H. and M. L. Littman. 1992. Interactions between learning and evolution. In: *Artificial Life II*, eds. C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, 478–507. Reading, MA: Addison-Wesley.
- Andrade, G., G. Ramalho, H. Santana, and V. Corruble. 2005. Extending reinforcement learning to provide dynamic game balancing. In: *Proceedings of the Workshop on Reasoning, Representation, and Learning in Computer Games, 19th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 7–12.
- Champandard, A. J. 2004. *AI Game Development*. New Riders Publishing.
- Cole, N., S. J. Louis, and C. Miles. 2004. Using a genetic algorithm to tune first-person shooter bots. In: *Proceedings of the 2004 Congress on Evolutionary Computation*, pages 139–145.

- Csikszentmihalyi, M. 1990. *Flow: The Psychology of Optimal Experience*. New York: Harper & Row.
- Demasi, P. and A. J. de O. Cruz. 2002. On-line coevolution for action games. In: *Proceedings of the 3rd International Conference on Intelligent Games and Simulation (GAME-ON)*, pages 113–120.
- Fogel, D. B. 1993. Using evolutionary programming to construct neural networks that are capable of playing tic-tac-toe. In: *Proceedings of the IEEE International Conference on Neural Networks*, page 875–880, San Francisco, CA, USA: IEEE Press.
- Fogel, D. B. 2002. *Blondie 24: Playing at the Edge of AI*. Morgan Kaufmann.
- Fogel, D. B., T. J. Hays, and D. R. Johnson. 2004. A platform for evolving characters in competitive games. In: *Proceedings of the Congress on Evolutionary Computation (CEC-04)*, pages 1420–1426, June 2004.
- Funge, J. D. 2004. *Artificial Intelligence for Computer Games*. Wellesley, MA: A. K. Peters Ltd.
- Gallagher, M. and A. Ryan. 2003. Learning to play Pac-man: An evolutionary, rule-based approach. In: *Proceedings of the Congress on Evolutionary Computation (CEC)*, pages 2462–2469.
- Graepel, T., R. Herbrich, and J. Gold. 2004. Learning to fight. In: *Proceedings of the International Conference on Computer Games: Artificial Intelligence, Design and Education*, pages 193–200, Reading, UK.
- Haynes, T. and S. Sen. 1995. Evolving behavioral strategies in predators and prey. In: *IJCAI-95 Workshop on Adaptation and Learning in Multiagent Systems*, pages 32–37, Montreal, Quebec, Canada: Morgan Kaufmann.
- Hunicke, R. and V. Chapman. 2004. AI for dynamic difficulty adjustment in games. In: *Proceedings of the Challenges in Game AI Workshop, 19th Nineteenth National Conference on Artificial Intelligence (AAAI'04)*, pages 91–96.
- Iida, H., N. Takeshita, and J. Yoshimura. 2003. A metric for entertainment of boardgames: Its implication for evolution of chess variants. In: *IWEC2002 Proceedings*, pages 65–72, Kluwer.
- Isla, D. and B. Blumberg. 2002. New challenges for character-based AI for games. In: *Proceedings of the AAAI Spring Symposium on AI and Interactive Entertainment*, pages 41–45, Stanford, CA, USA: AAAI Press.
- Johnson, S. 2004. *Adaptive AI*. Hingham, MA: Charles River Media.
- Kaiser, S. and T. Wehrle. 1996. Situated emotional problem solving in interactive computer games. In: *Proceedings of the VIXth Conference of the International Society for Research on Emotions*, pages 276–280, ISRE Publications.
- Kaiser, S., T. Wehrle, and S. Schmidt. 1998. Emotional episodes, facial expressions, and reported feelings in human computer interactions. In: *Proceedings of the Xth Conference of the International Society for Research on Emotions*, pages 82–86, ISRE Publications.
- Khoo, A. and R. Zubeck. 2002. Applying inexpensive techniques to computer games. *IEEE Intelligent Systems*, 17(4): 48–53.
- Koster, R. 2005. *A Theory of Fun for Game Design*. Scottsdale, AZ: Paraglyph Press.
- Koza, J. 1992. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press.
- Laird, J. E. 2002. Research in human-level AI using computer games. *Communications of the ACM* 3(8): 32–35.
- Laird, J. E. and M. van Lent. 2000. Human-level AI's killer application: Interactive computer games. In: *Proceedings of the Seventh National Conference on Artificial Intelligence (AAAI)*, pages 1171–1178, Saint Paul, MN.
- Lazzaro, N. 2004. *Why We Play Games: Four Keys to More Emotion Without Story*. XEO Design Inc., Technical Report.
- Lucas, S. 2005. Evolving a neural network location evaluator to play Ms. Pac-Man. In: *Proceedings of the IEEE Symposium on Computational Intelligence and Games*, pages 203–210, Cambridge, MA: Colchester, UK.
- Luke, S. and L. Spector. 1996. Evolving teamwork and coordination with genetic programming. In: *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 150–156, Palo Alto, California, USA: MIT Press.
- Malone, T. W. 1981. What makes computer games fun. *Byte* 6:258–277.
- McQuiggan, S., S. Lee, and J. Lester. 2006. Predicting user physiological response for interactive environments: An inductive approach. In: *Proceedings of the 2nd Artificial Intelligence for Interactive Digital Entertainment Conference*, pages 60–65.

- Miller, G. and D. Cliff. 1994. Protean behavior in dynamic games: Arguments for the co-evolution of pursuit-evasion tactics. In: *From Animals to Animats 3: Proceedings of the Third International Conference on Simulation of Adaptive Behavior (SAB-94)*, pages 411–420, MIT Press.
- Ponsen, M. and P. Spronck. 2004. “Improving adaptive game AI with evolutionary learning. In: *Proceedings of the International Conference on Computer Games: Artificial Intelligence, Design and Education*, pages 389–396, Reading, UK.
- Rabin, S. 2002. *AI Game Programming Wisdom*.: Charles River Media, Inc.
- Rani, P., N. Sarkar, and C. Liu. 2005. Maintaining optimal challenge in computer games through real-time physiological feedback. In *Proceedings of the 11th International Conference on Human Computer Interaction*, pages 184–192.
- Read, J., S. MacFarlane, and C. Cassey. 2002. Endurability, engagement and expectations. In *Proceedings of International Conference for Interaction Design and Children*, ACM Press.
- Richards, N., D. E. Moriarty, and R. Miikkulainen. 1998. Evolving neural networks to play go. *Appl. Intell.* 8(1):85–96.
- Rosca, J. 1996. Generality versus size in genetic programming. In *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 381–387, Palo Alto, California, USA: MIT Press.
- Spronck, P., I. Sprinkhuizen-Kuyper, and E. Postma. 2004. Difficulty scaling of game AI. In: *Proceedings of the 5th International Conference on Intelligent Games and Simulation (GAME-ON 2004)*, pages 33–37.
- Stanley, K., Bryant, B., and R. Miikkulainen. 2005. Real-time evolution in the NERO video game. In: *Proceedings of the IEEE Symposium on Computational Intelligence and Games*, pages 182–189, Colchester, UK.
- Stanley, K. and R. Miikkulainen. 2002. Evolving neural networks through augmenting topologies: *Evolutionary Computation* 10(2):99–127.
- Sweetser, P. and P. Wyeth. 2005. GameFlow: A model for evaluating player enjoyment in games. *ACM Computers in Entertainment* 3(3).
- Taylor, T. 2000. “Artificial life techniques for generating controllers for physically modelled characters.” In: *Proceedings of the First International Conference on Intelligent Games and Simulation (GAME-ON 2000)*, ■.
- Tesauro, G. 2002. Programming backgammon using self-teaching neural nets. *Artificial Intelligence* 134:181–199.
- Thurau, C., C. Bauckhage, and G. Sagerer. 2004. Learning human-like movement behavior for computer games, In: *From Animals to Animats 8: Proceedings of the 8th International Conference on Simulation of Adaptive Behavior (SAB-04)*, pages 315–323, Santa Monica, CA: The MIT Press.
- Verma, M. A. and P. W. McOwan. 2005. An adaptive methodology for synthesising mobile phone games using genetic algorithms. In: *Proceedings of the Congress on Evolutionary Computation (CEC-05)*, pages 528–535, Edinburgh, UK.
- Weizenbaum, J. 1966. ELIZA—a computer program for the study of natural language communications between men and machines. *Communications of the Association for Computing Machinery* 9:36–45.
- Wikipedia, the Free Encyclopedia. 2005. Pac-Man, [Online]. Available: <http://en.wikipedia.org/wiki/Pac-man>
- Woodcock, S. 2001. *Game AI: The State of the Industry 2000–2001: It's not Just Art, It's Engineering*.
- Yannakakis, G. N. 2005. *AI in Computer Games: Generating Interesting Interactive Opponents by the use of Evolutionary Computation*. Ph.D. thesis, University of Edinburgh.
- Yannakakis, G. N. and J. Hallam. 2004a. “Evolving opponents for interesting interactive computer games.” In: *From Animals to Animats 8: Proceedings of the 8th International Conference on Simulation of Adaptive Behavior (SAB-04)*, pages 499–508, Santa Monica, CA, USA: The MIT Press.
- Yannakakis, G. N. and J. Hallam. 2004b. *Interactive opponents generate interesting games*. In: *Proceedings of the International Conference on Computer Games: Artificial Intelligence, Design and Education*, pages 240–247, Reading, UK.
- Yannakakis, G. N. and J. Hallam. 2005a. A generic approach for generating interesting interactive Pac-man opponents. In: *Proceedings of the IEEE Symposium on Computational Intelligence and Games*, pages 94–101, Colchester, UK.
- Yannakakis, G. N. and J. Hallam. 2005b. A generic approach for obtaining higher entertainment in predator/prey computer games. *Journal of Game Development* 1(3):23–50.

- Yannakakis, G. N. and J. Hallam. 2005c. A scheme for creating digital entertainment with substance. In: *Proceedings of the Workshop on Reasoning, Representation, and Learning in Computer Games, 19th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 119–124.
- Yannakakis, G. N. and J. Hallam. 2006. Towards capturing and enhancing entertainment in computer games. In: *Proceedings of the 4th Hellenic Conference on Artificial Intelligence, Lecture Notes in Artificial Intelligence*, vol. 3955, pages 432–442 Heraklion, Greece: Springer-Verlag.
- Yannakakis, G. N. and M. Maragoudakis. 2005. Player modeling impact on player's entertainment in computer games. In *Proceedings of the 10th International Conference on User Modeling; Lecture Notes in Computer Science*, vol. 3538, pages 74–78. Edinburgh: Springer-Verlag.
- Yannakakis, G. N., J. Hallam, and H. H. Lund. 2006. Capturing entertainment through heart-rate dynamics in the playware playground. In *Proceedings of the 5th International Conference on Entertainment Computing, Lecture Notes in Computer Science*, vol. 4161, pages 314–317 Cambridge, UK: Springer-Verlag.
- Yannakakis, G. N., J. Hallam, and H. H. Lund. 2006. Comparative fun analysis in the innovative playware game platform. In *Proceedings of the 1st World Conference for Fun'n Games*, pages 64–70, Preston, UK.
- Yannakakis, G. N., J. Levine, and J. Hallam. 2004. An evolutionary approach for interactive computer games. In *Proceedings of the Congress on Evolutionary Computation (CEC-04)*, pages 986–993.
- Yao, X. 1999. Evolving artificial neural networks. *Proceedings of the IEEE* 87(9):1423–1447.

## NOTES

1. Further details of this strategy are presented in Yannakakis and Hallam (2004).
2. The case of  $I_1 = I_2$  is not investigated since the 1–2 pair is not taken into consideration and  $z'$  values cannot be computed for subjects that played that particular pair of sets.

# Difficulty Scaling through Incongruity

**Giel van Lankveld and Pieter Spronck**

Tilburg centre for Creative Computing  
Tilburg University, The Netherlands

**Matthias Rauterberg**

Designed Intelligence group  
Eindhoven University of Technology, The Netherlands

## Abstract

In this paper we discuss our work on using the incongruity measure from psychological literature to scale the difficulty level of a game online to the capabilities of the human player. Our approach has been implemented in a small game called *Glove*.

## Introduction

The entertainment value of modern day computer games is a topic of much interest. Iida, Takeshita, & Yoshimura (2002) created an entertainment measure for board games, but their measure uses concepts that have no equivalent in modern computer games. Yannakakis & Hallam (2005) attempted a similar approach for predator-prey computer games using ad-hoc calculations, with limited success. As a measure of interest (and thus entertainment) psychological literature introduced the concept of 'incongruity' (Rauterberg 1995). Incongruity is the difference between the complexity of an environment and the complexity of the mental model a human has of the environment. Research has shown that human interest in an environment is highest when the incongruity is neither too high nor too low. For software environments, indications for the complexity of the mental model can be derived from the human's interactions with the environment (Rauterberg 1995). Since the actual complexity of a game can be defined as its difficulty level, it seems possible to calculate incongruity, and thus entertainment value, during a gaming session. In our research, we developed a small computer game, named *Glove*, that uses the concept of incongruity to scale the difficulty level of the game online to the capabilities of the human player.

We first discuss incongruity as a measure for the player interest in a game. We then explain how we used incongruity in *Glove* to adapt game difficulty to the player's skill. Finally, we discuss future work.

## Incongruity for Game Complexity

When people encounter an environmental context, they need to process information about that context. They do this by using an internal mental model of the context. This internal

model, also called the 'system,' can be said to have levels of mental complexity in several dimensions. E.g., if the context is a game, then the system is the mental model that the player has of the game. This model may have, for instance, a tactical complexity, which describes how well the player is able to deal with game tactics, and an interface complexity, which describes how well the player is physically able to control the game.

Incongruity is defined as the difference between the environmental (context) complexity and the mental (system) complexity. There is positive incongruity when the environmental complexity is higher than the mental complexity and negative incongruity whenever it is lower. For instance, in the case of a game, a negative incongruity in tactical complexity would indicate that the human player's tactical understanding of the game is such that he is able to defeat the game easily. Figure 1 schematically visualises the concept of incongruity.

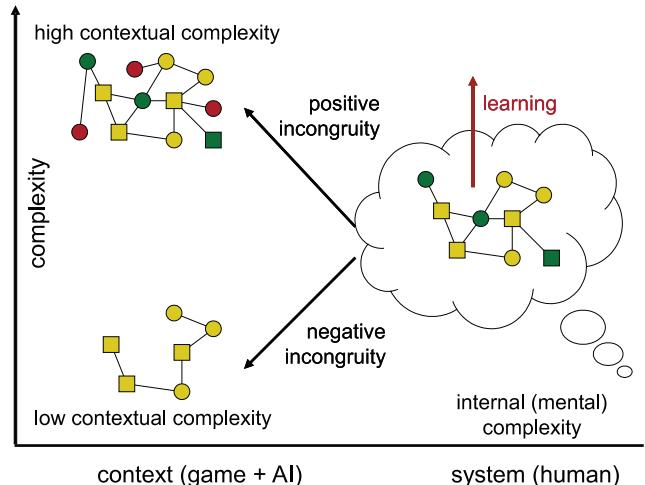


Figure 1: Incongruity.

In the case of positive incongruity humans have a tendency to optimise their mental complexity to the level of environmental complexity through seeking and exploration in order to match mental complexity to the environmental complexity. I.e., they learn. However, when they encounter

positive incongruity that is above a personal threshold or comfort level they tend to display avoidance behaviour. In situations of large negative incongruity humans start to look for new stimulation. (Rauterberg 1995). In the case of game playing, this means that when incongruity is large and positive, the player gets frustrated with the game and refuses to play. In contrast, when incongruity is large and negative, the player is bored.

For games, there is usually no direct way to measure mental complexity. The only possibility is to measure the complexity of the player's behaviour in a game and infer the mental complexity from that. Rauterberg (1993) states that low mental complexity will lead to the player's behaviour being largely determined by heuristics, while expert players, with high mental complexity, use other, more straightforward methods.

## Glove

The game *Glove*, displayed in Figure 2, is a simple, turn-based, side-scrolling arcade game, in which the player controls one character, which must be moved from the left side to the right side of the playing area. The player has just enough health to reach that goal. The player encounters enemies of three different tactical types, which may damage him, which costs health. However, defeating enemies gains the player health. Therefore, if the player encounters enemies which he finds easy to defeat, he will easily reach the goal. However, if the enemies are too difficult for him, the goal cannot be reached. A good balance of enemies will allow the player to reach the goal, but just barely.

*Glove* is an attempt to cause situations of perpetual positive incongruence within personal preferences of the player. We assume that the player finds a well-balanced game most interesting to play. By adapting the context complexity to the inferred mental complexity of the player, so that incongruity is at a constant, balanced level, the player should continually experience interest. This is also known as "flow" (Csikszentmihalyi & Csikszentmihalyi 1988).

During gameplay the amount of damage that each of the enemies inflicts on the player is measured. This defines the complexity of each enemy type, difficult enemies having a higher damage amount. The environmental complexity is defined as the total of the complexities of the different enemies. By increasing the amount of difficult enemies the environmental complexity rises. By decreasing the amount of enemies in general and the amount of difficult enemies specifically the environmental complexity drops.

The mental complexity of the player is represented by the ease of working through an area of the game. A constant score is kept of the progress of the player and the amount of damage he has sustained. If the amount of progress is larger than the amount of damage sustained then the player has a larger mental complexity than the game.

Incongruity is at a balanced level if the player maintains at all times just enough health to reach the game's goal, but not more.

*Glove* has three settings for incongruity: a hard one that makes the player lose after having progressed through about



Figure 2: Glove.

three-quarters of the game world, an easy one that lets the player reach the goal with plenty health to spare, and a balanced one as described above. Our assumption is that with the first two settings the player will get frustrated or bored, respectively, but that with the balanced setting the player is encouraged to learn, and will constantly increase his mental complexity.

## Future Work

In future work, we will research whether our assumptions on the effect of adapting to incongruity is as we assumed. If that is the case, then we will implement these concepts in other, more complex games.

## Acknowledgements

This research is supported by a grant from the Dutch Organisation for Scientific Research (NWO grant 612.066.406).

## References

- Csikszentmihalyi, M., and Csikszentmihalyi, I. 1988. *Introduction to part IV in optimal experience: Psychological studies of flow in consciousness*. Cambridge, UK: Cambridge University Press.
- Iida, H.; Takeshita, N.; and Yoshimura, J. 2002. A metric for entertainment of boardgames: Its implication for evolution of chess variants. In Nakatsu, R., and Hoshino, J., eds., *Entertainment Computing: Technologies and Applications*, 65–72. Boston, MA: Kluwer Academic Publishers.
- Rauterberg, M. 1993. Amme: an automatic mental model evaluation to analyze user behaviour traced in a finite, discrete state space. *Ergonomics* 36(11):1369–1380.
- Rauterberg, M. 1995. About a framework for information and information processing of learning systems. In Falkenberg, E.; Hesse, W.; and Olivé, A., eds., *Information System Concepts*, 54–69. IFIP Chapman & Hall.
- Yannakakis, G., and Hallam, J. 2005. A generic approach for obtaining higher entertainment in predator/prey computer games. *Journal of Game Development* 1(3):23–50.