Real-time Experience-Driven Procedural Content Generation: Thesis Advancement Proposal

Cameron Alston
UC Santa Cruz

January 25, 2015

Contents

Project Summary

Modern video game design and production is an iterative and complex set of processes that take an idea and turn it into a refined entertainment product. The process as a whole, from design conception to shipped title, can take many years and hundreds of millions of dollars to complete. A very large portion of the process of producing a video game comes in the form of user testing and design iteration. Typically the player testing process involves many users playing the game (in a pre-release state) under heavy supervision from designers so that particular aspects of the playability of the game can be exposed. The designers then use the information that they acquired during the testing process to iteratively change the game in the hopes of improving the overall game experience.

Recent research in this field includes many studies that involve data mining approaches to modeling players' actions and intentions in games. The models that are then developed are studied to attempt to reveal weaknesses (or strengths) in the game's design, opportunities for the designer's to implement a new feature, or to assess the player's experience. Once the data has revealed something to the designers the game can be modified and then tested again, in an iterative fashion. This process occurs offline (meaning that no changes to the game are implemented while the game is being played), and is quite similar to the previous work that was done in iterative game design approaches but with the addition of using data mining to assist the designer's observations.

Very few studies have been performed that attempt to utilize player models and game design principles to adjust game content in real-time. While some good work has been done, there is still a lot of room for more comprehensive and in-depth studies of the effects of introducing adaptively generated game content, and especially studies that attempt to introduce content during the execution of a game-level. It is the aim of this work to explore the field of experience-driven procedural content generation for player modeling and real-time content generation.

This proposal outlines a previously untried method of informing the design of a game in real-time using experience-driven procedural content generation. The study will show that real-time, adaptive game design methods are effective at improving the player's experience across several dimensions, including fun, challenge, flow, frustration, excitement, and relaxation. These results will be exemplified in a 3D dungeon crawler game environment, with player models being generated from comprehensive user studies using real player data.

Project Description

1 Introduction

1.1 Motivation

Video games have become a major part of today's society, with Americans spending over $21B USD in the United States last year with at least one dedicated video game console in over half of the homes in America and with 59% of Americans playing them [4]. It has been quite a long journey, with the complexity and level of detail that goes into them rising to new and greater levels with every new game that hits the market. Some of the first games that were published, 50 years ago, were considered marvels of technology just to have incredibly simple interactions with a controller and a few pixels moving on the screen. We now have games that cost hundreds of millions of dollars and take several years to produce, that are built to engage and connect players for hundreds of hours and across the globe. These games are fully immersive with millions of pixels being rendered in 3D at upwards of 60 frames per second, and the technology has been growing every day.

A large part of the multi-million dollar process of producing a video game is in testing and iteratively improving the design of the game. During this part of the process, designers very carefully observe players playing their games, logging their behaviors and utilizing vast amounts of gameplay data to gain information about the way their game was experienced. Not just is the complexity of the games being created growing at an incredible rate, but the types and amount of new players that are playing today's games is increasing. It is becoming increasingly difficult for game testing to effectively encompass all of the possible responses to a games content so that a game's design can be adequately informed prior to being released to the public.

Given the increasing development cost and complexity of the games that are being produced, and the diversity in skill, background, and personal preferences of the ever increasing gamer audience, it is intuitive to think that there is a desire for games that are able to entertain and connect with as many players as possible and on as many levels as possible. Recent academic research in the area of player modeling and procedural content generation has opened up a new area for automatic and personalized generation of game content, allowing the design of the game to be flexible and for the players' experience to be unique to each different player.

It is the aim of the work proposed here to extend the findings of previous research in the area of experience-driven procedural content generation (ED-PCG) to improve upon the previously explored methods by introducing generated content in real-time during the execution of a game level, and to validate the ideas that game design and player experience can be personalized using ED-PCG methods.

1.2 Research Questions

The primary aim of this research is to study whether experience-driven procedural content generation is useful for improving player experience in games. There are many facets to evaluating whether ED-PCG improves a player's experience, so, more specifically, this research will aim to answer the following questions about player experience and ED-PCG.

- **RQ1**: Does player experience modeling accurately reflect the experience the player while they are playing a game?

This research question will be answered through the construction of the models outlined in Section 4.3.

- **RQ2**: Does player experience modeling give enough information about game content to inform the game's design in real time?

This research question will be answered through the results of the training of the neural network in 4.3.2.

- **RQ3**: Does procedurally generating game content in real-time via player experience modeling have the intended effect?

This research question will be answered in after the results of the user studies proposed in Section 4.6

2 Related Work

Previous work in the area of adaptive game design encompasses work that has been done in several fields, as adaptive game design incorporates concepts and methods that previously have been thought of as fields all to themselves.

In order for an adaptive game design system to provide a personalized experience to a player, the system must have a notion of what the player consistently does. A great deal of research has been done in the area of player modeling towards the effect of using data to understand what players do/see/understand while playing a game.

Also tied into understanding and predicting what the player wants to see next in the game is understanding how to evaluate what the user is experiencing when they play a game. Player modeling and player experience research are very closely tied together in that they are both used to understand the player's behavior and how it relates to what they experience when they play a game.

Another requirement for an adaptive game design system to be feasible is to understand how games are fundamentally designed. Much of game design is done by designers using methods and instincts that they have acquired and developed over the course of their careers to know what is good to do when making a game. Many times they will not even be able to tell you why they included a particular aspect at a particular point, just that they instinctually 'knew it would be good.' Design patterns in games research aims to understand what aspects of a game's design are common within genres, and what effect these aspects have on the players' experiences in games.

Experience-driven procedural content generation (ED-PCG) is an extension of the procedural content generation research field, and is a natural extension of player experience, design pattern, and player modeling research. In ED-PCG, systems are developed that allow for personalized game content to be displayed to the player, using models of players' behavior to achieve a more robust player experience.

2.1 Player Experience Modeling

Player experience modeling (PEM) is a field of research dedicated to creating computational models of players in games, including the detection, modeling, prediction and expression of human player

characteristics which are manifested through cognitive, affective and behavioral patterns. Work has been done in defining player modeling from a general standpoint, and many studies that use player models for inferring information about player behavior have been conducted.

Smith, et al., developed a taxonomy for classifying player modeling approaches. The authors' taxonomy can be seen in Table 1.

Domain	**Purpose**	**Scope**	**Source**
Game Actions *details recorded inside of the game's rule system*	Generative *details observable in the player as a result of play*	Individual *applicable only to one player*	Induced *learned/fit/recorded by algorithmic means*
Human Reactions *literally produces details in place of a human player*	Descriptive *conveys a high-level description, usually visually or linguistically*	Class *applicable to a subpopulation*	Interpreted *concluded via fuzzy/subjective reasoning from records*
		Universal *applicable to all players*	Analytic *derived purely from the game's rules and related models*
		Hypothetical *unlikely to be applicable to any players, but interesting nonetheless*	Synthetic *justified by reference to an internal belief or external theory*

Table 1: Facets of player modeling taxonomy and their possible values [22].

In their book chapter, Yannakakis, et al., summarized the core components of the player modeling research field. Their generalized definitions include both model-based and model-free approaches to the problem, identifying the core aspects of data input and model output that are common among traditional player modeling approaches [30]. A digram of their findings can be seen in Figure 1.

Another study that employed player experience modeling techniques similar to those proposed in this study is by Yannakakis, et al., in their study of affective camera control in games [29]. In their work, the researchers utilized biometric input in the form of heart rate and skin conductance monitors, in addition to subjective user tests to draw inference between the position of the camera in the game, physiological response, and affective state. The results of the work were not able to produce a good classifier, averaging about %55 accuracy for all of the affective dimensions. An image of the game environment that was used in their study can be found in Figure 2.

One study that is most closely related to the player modeling approach adopted in this work is by Pederson, et al., where they used gameplay based features and player surveys to create affective models of game content [20]. The game environment that the researchers chose was a procedurally generated platforming game called Infinite Mario by Markus Persson, which is based off of the classic Nintendo game Super Mario Bros. and an image of the game environment that was used in their study can be seen in Figure 3.

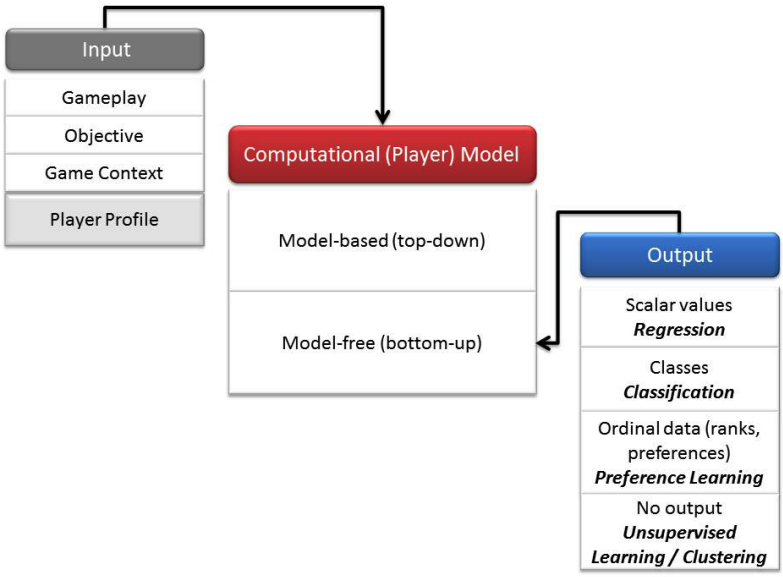


Figure 1: High level view of player modeling approaches [30]

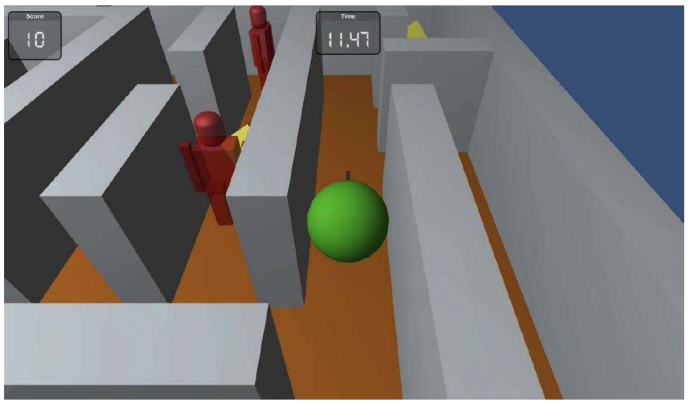


Figure 2: The game testbed environment used in [29]

Parameters of the game that could be modeled included 4 different features:

- **Number of gaps**: the number of gaps in the level.
- **Gap width**: the average gap width.
- **Spatial diversity of gaps**: the entropy in the width of the gaps in the level.
- **Direction switch**: the percentage of the level played moving to the left.

The results of the study showed that the player modeling approach was able to predict with relatively high accuracy. The researchers were able to get up to >90% accuracy for some of the affective dimensions being studied, but their studied lacked a sufficient amount of participants needed to create models capable of representing all of the affective states with high accuracy.

Figure 3: The game testbed environment used in [20]

2.2 Procedural Content Generation

Procedural content generation (PCG) research is work that is concerned with the creation of content automatically, through algorithmic means. PCG methods can be classified in to several categories [25]:

- **Online vs. offline**: denotes whether PCG content is generated during gameplay or in between gameplay sessions.
- **Necessary vs. optional**: denotes whether PCG content is imperative for the player to progress in the game (i.e. level geometry) or for other reasons like aesthetics or narrative.
- **Random seed vs. parameter vector**: denotes whether PCG content is generated randomly or via specified parameters. Can be be somewhere in between with some randomization of parameters in a parameter vector.
- **Stochastic vs. deterministic**: denotes whether PCG content is generated exactly the same given the same parameter vector or random seed
- **Constructive vs. generate and test**: constructive algorithms work from the ground up generating content and have no ability to replace content that is already generated. Generate and test algorithms generate content and test for feasibility, and when some content is infeasible it can be removed and replaced.

In the realm of games, PCG refers to the ability to affect game content that is not NPC behavior or the engine itself. Content that can be affected includes terrain, maps, levels, stories, dialogue, quests, characters, rulesets, camera profiles, dynamics, and weapons [31]. Procedural content generation research has been around for a considerable amount of time, and has had some great successes in the professional game industry, but is still not as widely adopted as researchers would hope. There have been numerous studies in PCG, but the areas of PCG that are most

applicable to the work being proposed in this study are those of search-based procedural content generation and experience-driven procedural content generation.

2.2.1 Search-based Procedural Content Generation

Togelius, et al., first introduced the term search-based PCG in their paper of the same name as the research subfield [25]. Search-based procedural content generation (SB-PCG) is a special case of the generate-and-test approach to PCG, with the following qualifications:

- The test function does not simply accept or reject the candidate content, but grades it using one or a vector of real numbers. Such a test function is sometimes called a fitness function and the grade it assigns to the content its fitness.
- Generating new candidate content is contingent upon the fitness assigned to previously evaluated content instances; in this way the aim is to produce new content with higher fitness.

2.2.2 Experience-Driven Procedural Content Generation

Experience-driven procedural content generation (ED-PCG) is the field of research that most closely fits with the work proposed in this document. It is a field of game research dedicated to understanding what a player wants to see in a game at a particular time, and presenting it to them. The term was first introduced in a paper by Yannakakis, et al., where the authors summarized work in affective game development and talked about how cognitive modeling coupled with real-time adjustment of game content toward individual user needs and preferences could be a important step in the development of effective and meaningful PCG research [31].

Experience-driven procedural content generation combines elements of player experience modeling and procedural content generation research and extends them to provide systems that allow real-time augmentation of game content that is directed toward a personalized experience for the player. Research in the field has only been going on for a couple of years now, and as such there are not many practical applications of the methods that have been developed.

Yannakakis, et al., first developed a taxonomy of the ED-PCG research field. In their approach, content is viewed as the building blocks of games, and games as potentiators of experience. Content can be seen as indirect building blocks of player experience, and as such, must be evaluated on the quality of the experience they generate. Once an appropriate evaluation metric is developed, potential content can be searched through and optimized over to maximize the experience for the player. The components of ED-PCG are [31]:

- **Player experience modeling**: player experience is modeled as a function of game content and player (the player is characterized by her playing style, and her cognitive and affective responses to gameplay stimuli).
- **Content quality**: the quality of the generated content is assessed and linked to the modeled experience of the player.
- **Content representation**: content is represented accordingly to maximize efficacy, performance and robustness of the generator.
- **Content generator**: the generator searches through the content that optimizes the experience for the player according to the acquired model.

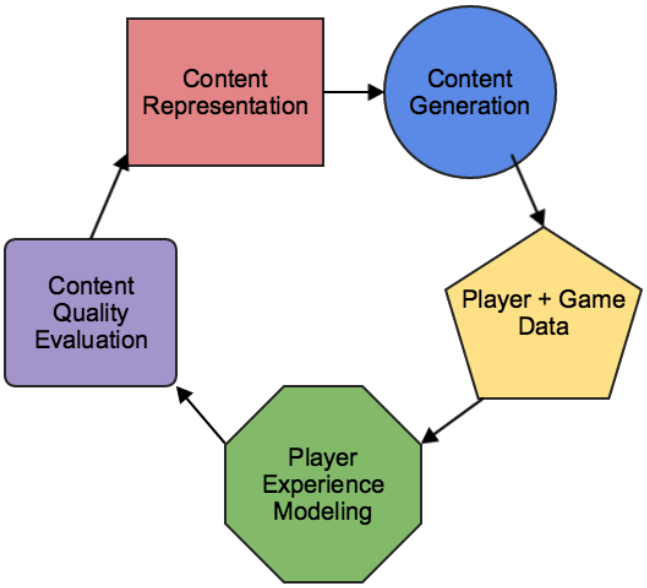
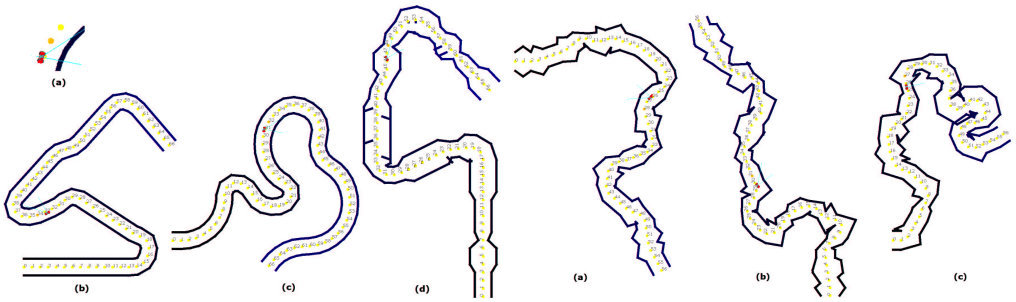


Figure 4: ED-PCG model [31]

Togelius, et al., introduced one of the first adaptive game design systems, using models of player behavior and generalized metrics of player entertainment a a fitness function for evolving personalized tracks with evolutionary algorithms in a simple racing game [26] [24]. The system used theoretical agents modeled after players of varying skill levels. The skill levels were represented by tendencies to maintain a higher speed, avoid collisions with walls, and make progress through tighter turns. Several tracks were hand-crafted to fit the model of each of the player types. Once the player models were defined, an evolutionary preference learning algorithm was used to tune tracks to best fit the preferences of each player type. The tracks can be seen in Figure 5 with (a), (b), and (c) representing the bad, average, and skilled player types.



(a) Hand-crafted tracks. (b) Evolved tracks.

Figure 5: Race tracks used in [26]

Their findings showed that it was possible to model players' behavior and was promising towards making a system that is tailored to a specific player's preferences. Their results were limited in the fact that they weren't able to actually test the system with real players. The researchers had 5 real human test subjects play the racing game and noticed that the play styles of the players fit into

one their theoretical models with sufficient accuracy, and as such they concluded that their models were accurate enough to model human players. They also did not attempt to change any of the designed levels in real-time. Their work was primarily to show that a level could be automatically designed with a specific player's preferences in mind, but required knowledge of the specific player's player type beforehand in order to match them up with an appropriate level.

Liapis, et al., developed models for producing personalized visually aesthetically pleasing 2D space ships in a game environment. Their approach followed a two-step methodology, wherein they take existing game content and user preference ratings in the game to evolve spaceship models that are more visually appealing to the player [14].

An example of some of the spaceship models that their system generated can be seen in Figure 6. The system that the authors developed was able to optimize the spaceships content across 7 dimensions. The spaceships shown in the Figure 6 are examples that optimize toward one of each of the feature dimensions.

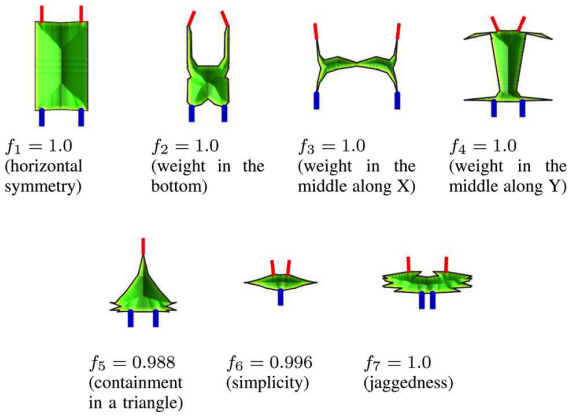


Figure 6: Single-dimension optimized spaceships [14]

3 Previous Work

Prior to this work, the PI was investigating methods of automating camera control in game environments using gaze data as a primary input for developing models of player and camera behavior. The primary work that the PI accomplished was in designing a neural network based system for automating the control of the camera in a first-person shooter game [3].

The work is relevant to the work proposed in this paper because it utilizes a neural network based approach to modeling player camera control and adapting the camera's viewpoint in real-time. The inputs to the model were gameplay data and physiological objective data in the form of gaze position on the screen.

Additionally, there was a user study performed that was designed to gauge the players' thoughts and frustrations regarding the automated camera control system, of the purpose of improving the feature generation

3.1 Neural Network Construction

Many features were calculated during the gameplay sessions in the case study, including:

- **Time watching enemies**: the amount of time the player focuses on enemies.
- **Time watching HUD**: the amount of time that the player was focused on elements of the HUD.
- **Time watching pickup items**: the amount of time the players looked at items they can interact with.
- **Time searching for targets/items**: the amount of time that the player spent looking around the environment/not focusing on anything for an extended period of time.
- **Camera speed**: the relative speed of the camera, $S = \Delta Camera(\theta, \phi)/T$ where S is speed and T is time. θ and ϕ are the polar and azimuthal angles. Relative speed is computed over the previous 1 second of game play.
- **Number of weapons picked up**: the amount and type of weapons acquired.
- **Number of health packs picked up**: the number of health recovery items picked up.
- **Average health**: the player's average health over the past 10 seconds.
- **Number of enemies spawned**: the number of enemies that the player triggered to appear.
- **Number of enemies killed**: the number of enemies that the player defeated.
- **Firing accuracy**: The number of targets hit versus the number of bullets fired.
- **Time with enemies in front**: the amount of time that players spent with enemies in vision.
- **Time with enemies in rear**: the amount of time that players had enemies primarily behind their front facing vector.
- **Time surrounded by enemies**: the amount of time that the players were surrounded (enemies evenly spaced around the player) by enemies.
- **Average enemy distance**: the distance that the player stayed away from active enemies.

3.2 User Study

As a first step to the training of the learning algorithm for controlling the camera, a preliminary user study was conducted. The purpose for the user study was two-fold, to assess the usability and interest in such a system, and to acquire some initial gaze and gameplay based data for use in training the learning algorithm.

A beta-testing methodology was utilized for conducting user tests. A playable version of the gaze controlled camera system available for users to play, and surveys were conducted both before and after they played two different versions of the same game for 3 minutes each. One version of the game used a standard WASD and mouse-look FPS control system, and the other was using gaze to control the camera with WASD movement.

All except for one of the participants expressed interest in using the gaze controlled technology again, if the bugs of the system were worked out and the control scheme fine tuned more. In

particular, the majority of the users primary complaints were that were not able to accurately aim during gameplay and that they felt as though they were "chasing" the reticle around the screen with their gaze, causing them to get dizzy and experience eye fatigue. These effects were somewhat to be expected since the camera moves exactly as their eyes move with accuracy only verified within approximately 60 pixels from where the user is actually looking. As they attempted to focus on their intended target and the reticle only came within the 60 pixels, they tried to focus harder and harder to get the reticle to be exactly where they intended. Given the level of responses by the users toward the things that they liked and their interest in using the technology further, it seems as though it was a positive experience and their dissatisfaction with the accuracy of the camera and system will be ameliorated with further testing, development, and with the integration of a smarter camera control algorithm.

3.3 Continuing Work

The results from the eye tracking camera control study are still in progress. The paper was well received at AAAI 2014 but there are still many implementation details to be completed before further results can be reported.

3.3.1 Improving Features

The current system utilizes the gaze data streamed from the eye tracking device as an empirical location of focus in the game, but with more intelligent algorithms, such as one employing techniques similar to the one developed by Vidal et al. in their Pursuits application [27] and Koulieris, et al., in their study on saliency in virtual environments [13], it could be possible to more accurately and intelligently identify what the player's focus of attention is on. This improvement will enable the learning algorithm to be more effectively tuned as to salient elements of the environment.

4 Proposed Methods

This section outlines the proposed work to be done. It is broken down into several sub-sections, outlining each step in the process of eventually creating a system that will be able to answer the research questions stated above. The first section outlines the test environment, an open-source game. The second outlines the process of creating player models in the game, which leads to the third section where player behavior will be tied to affective properties of game content. In the fourth section, player models and affective models of game content will be brought together to adaptively generate game content, creating personalized game experiences through experience driven content generation. In the fifth section, methods for evaluating the effectiveness of the experience-driven procedural content generation are outlined.

4.1 Test Environment Overview

The test environment for the experiments will be an adventure dungeon crawler built in the Unity game engine. The environment was chosen for it's flexibility in producing procedurally generated dungeon environments that are controllable. The environment was also chosen for it's flexibility in the ability to choose and expand on game content, being that the Unity Asset Store provides a great deal of content at a very low cost.

4.1.1 Level Generator

The dungeon map generation for the experiments will be handled with a Unity plugin called Pro-D Procedural Dungeon Generator. The plugin allows for nearly limitless generation of dungeon levels that can be customized and tuned with many different variables and parameters. Some of the parameters that are available for customizing levels are outlined in Table 2. The real flexibility of the map generator is in the customization options within each type of generator. Options for several of the relevant map types are shown in 3. An image of the web interface for Prod-D can be seen in Figure 7.

Option	Description
World width	number of maps generated horizontally
World height	number of maps generated vertically
Map width	number of tiles each map has horizontally
Map height	number of tiles each map has vertically
Theme	defines the visual assets used for the map
Generator	defines the type of script used to create the map

Table 2: Options available in the Prod-D dungeon generator.

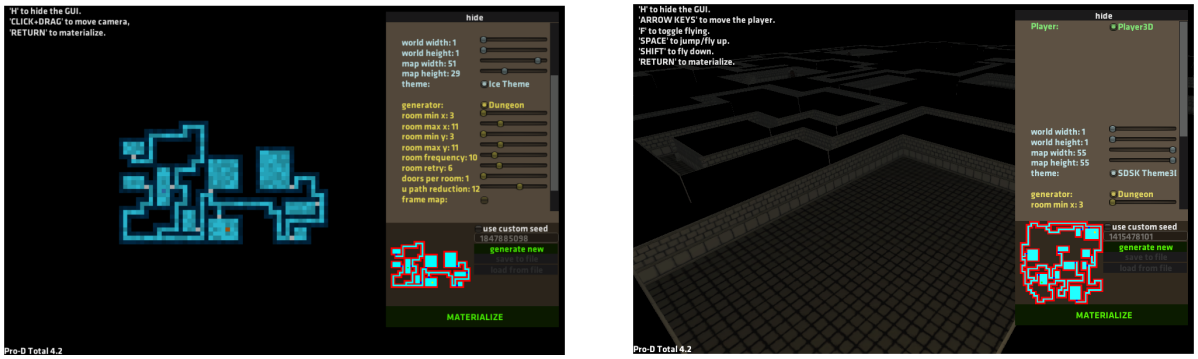
Options	Descriptions
Room min/max x/y	minimum and maximum width and height of any room in the generated map
Room frequency	amount of attempts to place a room on the map
Room retry	amount of retry attempts to place a room upon failing to place a room
Doors per room	amount of doors every room will have
U path reduction	amount of times corridors are simplified during map generation
Doors per room	amount of doors every room will have

Table 3: Options for the 'Dungeon' map type in the Prod-D dungeon generator.

The level generation algorithm roughly follows the methods presented by a game developer known as Captain Kraft in his web-based tutorials [8]. The algorithm that is used first determines where to place rooms of the desired sizes without rooms overlapping in the desired world size for the dungeon. Conflicting rooms are re-placed until a feasible layout is found or until the re-placement threshold has been reached. An image of the process can be seen in Figure 8.

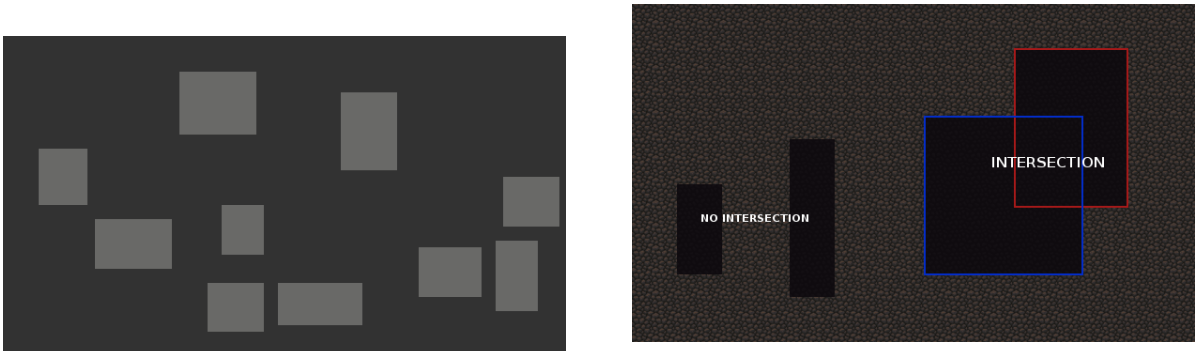
Once the rooms are placed, they must be connected with corridors. Using a similar approach to room placement, corridors are iteratively placed and tested for feasibility with the current map. The corridors are grown outward in vertical and horizontal directions from the randomly placed door locations for each room similarly to the images seen in Figure 9 but in the Pro-D system there is a parameter for corridor simplification that allows for more complex paths.

The final dungeon map looks something like what is seen in Figure 10.



(a) 2D. (b) 3D

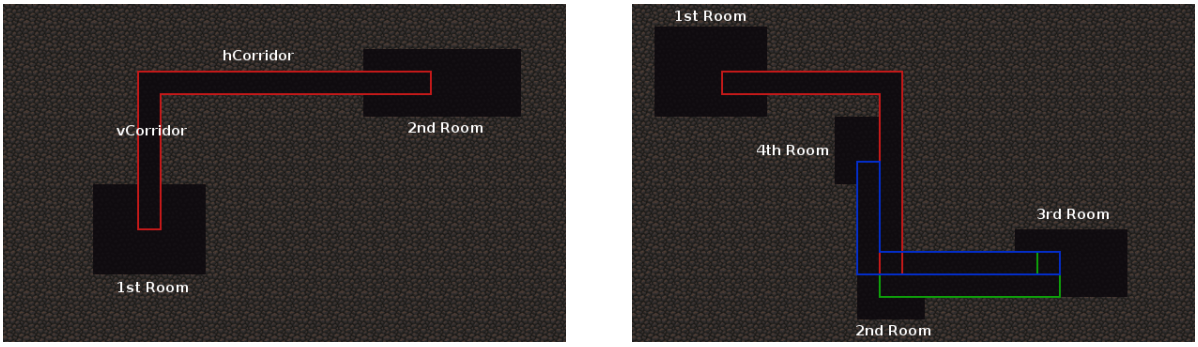
Figure 7: Pro-D dungeon generator interface. [23]



(a) Room placement.

(b) Rooms intersecting then recreate both and attempt to place again.

Figure 8: Dungeon room placement approach. [8]



(a) Corridor placement. (b) Corridor simplification opportunity.

Figure 9: Dungeon corridor placement approach. [8]

4.1.2 Player Type

The character that the players of the game will be able to use is an adventurer hero who wields a sword and a shield. He is equipped with several different attacks and animations that he can use

Figure 10: Simple generated dungeon. [8]

to negotiate and battle his way through the corridors of the dungeons.

Figure 11: Model of the player to be used in the game. [11]

4.1.3 Enemy Types

The characteristics of the 10 basic enemy types and 2 boss enemy types in the dungeon game environment are outlined in Table 4 and an image showing the models for the enemy characters can be seen in Figures 12 and 13. There are 5 basic enemy types with 2 variations for each type. The purpose of the enemies in the game is to provide a scalable level of challenge that will be determined through player modeling.

Enemy	Attack Type	Damage	Health	Speed	Difficulty
Goblin	Melee	Low	Low	Fast	Easy
Orc	Melee	High	High	Slow	Medium
Skeleton Archer	Ranged	Low	Low	Fast	Medium
Zombie	Melee	High	High	Slow	Hard
Vampire	Ranged	High	High	High	Hard

Table 4: Basic enemy descriptions.

Figure 12: Fantasy Horde enemy pack. [15]

(a) Manticore boss enemy.

(b) Golem boss enemy.

(c) Goblin Shaman boss enemy.

Figure 13: Boss enemies. [2] [1] [19]

Enemy	Attack Type	Damage	Health	Speed
Manticore	Flying Ranged	Highest	Highest	Fast
Golem	Melee	Highest	Highest	Slow
Skeleton Wizard	Lowest	Lowest	Slow	

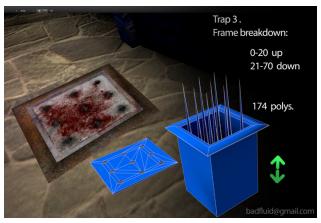
Table 5: Boss enemy descriptions.

4.1.4 Item Types

There will be both interactive and non-interactive items and pseudorandomly placed in the dungeon environment. Their purpose is both to enhance the visual aesthetics and interactive possibilities of the game levels. An image of the content can be seen in Figure 14. The chest item will provide a way for the player to gain treasure, or health bonuses.

(a) Various decorative items. [7]

(b) Chest item [11].



(c) Trap [5].

Figure 14: Various interactive and non-interactive items in the game environment.

4.1.5 Level Flow

The player will begin in a room and a cutscene will begin. The Goblin Shaman will appear and begin to describe the player's fate, being dropped into a dungeon maze with the only option being to fight his way out. The default first dungeon room will appear, with a tutorial style walkthrough describing the hero's attack actions and showing a chest that can be interacted with. The game then progresses with the player battling his way through the dungeon. At several points during the game's execution the Goblin Shaman will interrupt the player to inform him that the dungeon is morphing to make the hero more challenged (a disguise for the adaptive generation algorithm to create personalized content).

The level generation tool will have built into a metric for progression, highlighting start, end, mid-boss fight, and end-boss fight game events. In addition to these events, various rooms with treasure and basic enemy fights and corridor walking will be incorporated into each level's design, to contribute to the flow of the level, as described in [10].

4.2 Player Experience Modeling Overview

In this work, we propose a hybrid player modeling technique, that incorporates gameplay and subjective based PEM methods. Before the proposed methods can be outlined, the terminology of PEM is defined. There are several different approaches to PEM that existing research studies in the area fall into, subjective, objective, and gameplay based PEM. Key aspects of each approach are outlined below, with the space of possible PEM approaches including any combination of the 3 different methodologies.

- **Subjective based PEM**: refers to the self reporting of player experience through either free-response or forced data collection retrieved through questionnaires.
  - Free-response subjective reporting allows the player to report back their affective states without constraining their answers to any particular format. Free-response reporting can provide richer information about the player's affective state but can be much harder to analyze than forced reports.
  - Forced response subjective reports consist of questions that come in the form of multiple choice, check-boxes, rankings, preferences, or other short and clear question-answer

formats. Forced reports allow for much more organized data collection but are limited in their ability to acquire information that is not available from within the bounds of the questionnaires.

- **Objective based PEM**: incorporates access to multiple modalities of player input for the purpose of modeling the affective state of the player during play [31]. The types of player input that are considered are generally physiologically based, such as galvanic skin response (GSR), electrocardiography (ECG), electroencephalography (EEG), or gaze data. Objective PEM approaches can either be model-based or model-free, or some hybridization of the methodologies.
  - Model-based objective PEM refers to emotional models that are grounded in emotion theory and in which bodily functions are mapped to specific emotional responses. An example of such an approach is the mapping of increased heart-rate to high arousal and therefore player excitement.
  - Model-free objective PEM refers to the construction of an unknown mapping (model) between modalities of player input and an emotional state representation via user annotated data [31]. Classification and regression machine learning techniques are typically employed along with user reported affective states to draw correlations between physiological behavior and affective state.
- **Gameplay based PEM**: operates under the assumption that player actions and real-time preferences are related to player experience since games may affect the player's cognitive processing patterns and cognitive focus. Inversely, cognitive processes may influence emotions and the player's emotional state may be inferred by analyzing patterns of the interaction and associating user emotions with context variables [31]. Similarly to objective based PEM, gameplay based PEM approaches can be classified as either model-based or model-free, or hybridized, and the inputs to a gameplay-based player experience model are statistical spatiotemporal features of game interaction.
  - Model-based gameplay PEM approaches are typically inspired by a general theoretical framework of behavioral analysis and/or cognitive modeling but there are also theories about user affect that are specific to games. [31]
  - Model-free gameplay PEM approaches are concerned with the construction of an unknown mapping (model) between modalities of game interaction and emotional state representation via mappings between in game behavior and cognitive states like attention, challenge, and engagement. Classification and regression machine learning techniques are typically employed to draw correlations between physiological behavior and affective state.

The study outlined further in this section will utilize pairwise preference ratings for assessing players' subjective affective state while playing a game. The features computed will be analyzed using a model-free methodology and attributed to affective state to create a model of player experience. Additional gameplay-based features will be calculated and figured into the model using a hybrid model-free/model-based approach.

A diagram showing the PEM structure to be used in this study can be seen in Figure 15. The orange cells represent the continuous range between model-based and model-free objective

and gameplay based PEM approaches, with white cells representing the discrete options available for subjective modeling approaches. The blue outlines represent the methodologies that will be employed in the work proposed here.

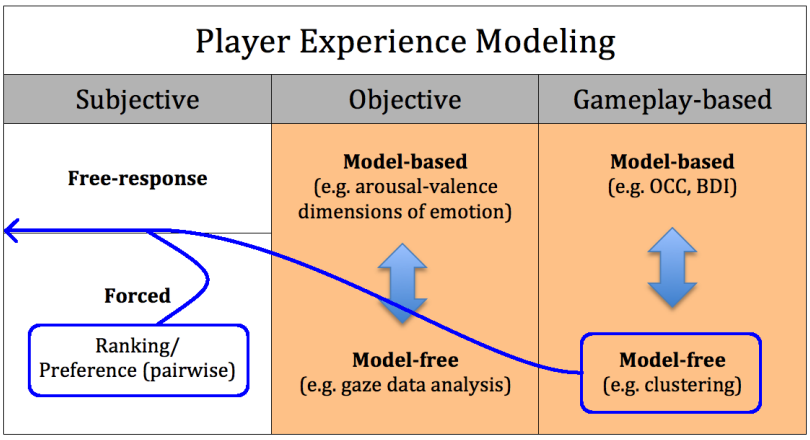


Figure 15: Diagram of PEM taxonomy and where this study fits in among the types of PEM approaches.

A flowchart depicting the flow of data in the processes that make up the PEM approach to be used in this study is shown in Figure 16. Each of the modules in the flowchart represents a process that is further detailed in this section.

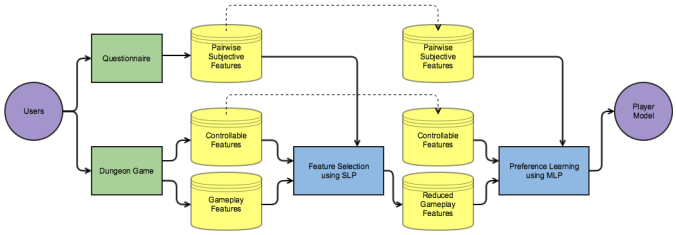


Figure 16: Flowchart of the PEM methodology used in this study.

4.2.1 Controllable Features

Controllable features of the game are those which relate to the generation of the level including the layout of the rooms, difficulty, and items. Some of the controllable features of the game that this study will be interested are listed in Table 6.

4.2.2 Subjective Features

In order to acquire the subjective data that is to be used in constructing the models of player behavior, user studies must be performed. The methodology proposed in this study is that of pairwise preference modeling, where players play a series of pairs of different short game levels and

Feature	Description
Level length	the number of rooms in the level.
Level topology	the difference between the size of the largest and smallest rooms and corridors in the level.
Enemy variety	how many different enemies types occur in the level.
Enemy frequency	how many enemies are in each room.
Boss fights	whether or not the player will have to fight either or both of the bosses in the level.
Trap frequency	relates to the number of traps in the level.
Chest frequency	relates to the number of treasure chests in the level.
Health frequency	relates to the number of health packets to be found in the level.

Table 6: Controllable features of the dungeon game levels.

compare the two experiences. Following the methodology in [28] and [32], a flowchart depicting the process of the individual user tests is shown in Figure 17.

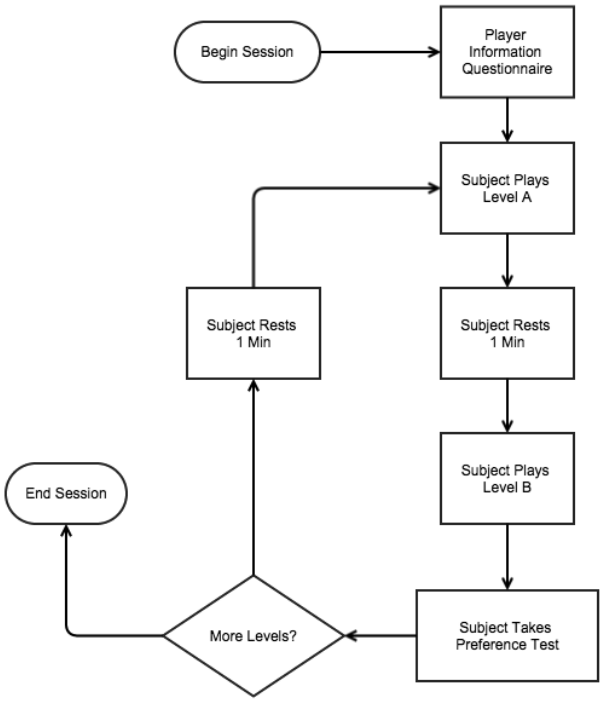


Figure 17: Diagram depicting the flow of the user tests.

The criteria that the player will be rating the the levels on are listed in the Table 7. The possible states affective states that we are interested in are chosen for their relevance in previous works [29], [10].

The pairwise preferences will be obtained through the use of a 4-AFC (alternative forced choice)

Affective State	Description
Fun	Describes a feeling of enjoyment from playing the level.
Challenge	Describes a feeling the player experienced in completing the level.
Flow	Describes a feeling of continuity from the sequence of events in the level.
Frustration	Describes a feeling of irritation or annoyance that arose form playing the level.
Excitement	Describes a feeling of thrill or elevated exhilaration from playing the level.
Relaxation	Describes a feeling peacefulness or calming from playing the level.

Table 7: Subjective preference categories.

protocol. Under this method, the user expresses preferences in the form:

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

A and B represent the different versions of the game level and E is one of the emotional states defined in Table 7.

To maximize the amount of subjective data collected, an instance of the game will be hosted on a server and data collected over time.

4.2.3 Gameplay Features

To collect gameplay data for the proposed experiments, telemetry methods will be programmed into the dungeon game environment. The telemetry will afford the calculation of many gameplay based features about player behavior in the dungeon environment. The features are organized by the categories Combat, Item, Movement, Time, Death, and Miscellaneous. The features are listed in Table 8.

4.3 Modeling Methodology

The approach to modeling the player's experience will follow a 2-phase approach, similar to the approach in [20]. An overview of the procedure used in [20] can be seen in Figure 18. The approach proposed in this work is similar to that used in [20] but in this work more feature selection methods will be implemented and compared, incorporating the latest and most novel techniques so as to achieve the highest potential accuracy of the model. The same 2-stage method, utilizing a single-layer perceptron (SLP) neural network to model the affective states of the players in relation to the gameplay features and a multi-layer perceptron (MLP) neural network on the reduced feature set output from the SLP neural net to achieve an accurate model of player preferences in relation to the features.

Category	Feature
Combat	% enemies killed % enemies killed (each type) % attacks landed % attacks dodged total damage taken total damage to enemy
Item	% treasure opened % health collected % vases smashed
Movement	% areas visited # jumps % time running % time ducking % traps avoided
Time	time to complete level average time per room average time per hall % time fighting % time fighting % time exploring
Death	total deaths deaths from enemy (each type) deaths from boss deaths from trap

Table 8: Gameplay based features.

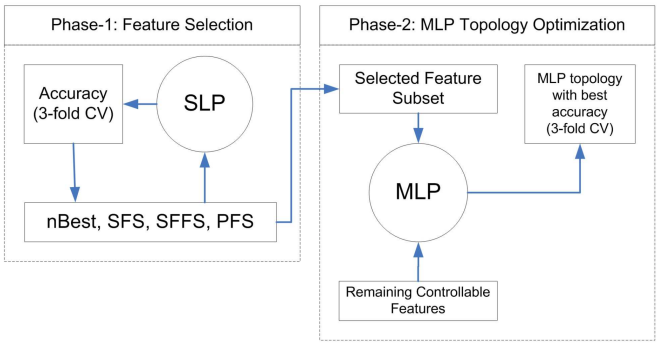


Figure 18: SMP and MLP feature selection and model training overview [20].

4.3.1 Feature Selection

Once the gameplay features have been logged, several feature selection methods will be employed and compared for determining the most relevant features in the player models. Feature selection is

used to cut out the features in the feature space that are unnecessary, or don't actively contribute, to modeling a player's behavior. This technique improves the human readability of the results, as it allows for more simple visualization of the relationships between features and affective state, and has also been shown to improve the learning quality for evolutionary training of the neural network.

The feature selection algorithms that will be implemented and compared include the n-best individual feature selection (nBest), sequential forward selection (SFS), sequential floating forward selection (SFFS), perceptron feature selection (PFS), and genetic feature selection algorithms (GFS). All feature selection algorithms will be run in the WEKA machine learning environment.

The 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 [32].

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.

Several studies have demonstrated the benefits of sequential floating search over standard sequential search. Floating search algorithms can be classified into forward and backward search. The sequential floating forward search (SFFS) algorithm performs the sequential steps of the SFS algorithm. However, each time an SFS step is performed, SFFS checks whether the performance function value can be increased if a feature is excluded from the current feature subset (i.e. one step of sequential backward selection is performed) [32].

Perceptron feature selection (PFS) is an aggressive-search variant of neural pruning and sequential backward selection. Rosenblatt's perceptron is used as a methodology for selecting appropriate feature subsets [21]. The algorithm to be used is similar to the algorithm used in [32], where 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:

1. Use artificial evolution to train the perceptron on the pairwise preferences, as in Section 4.3.2. Performance of the perceptron is evaluated through 3-fold cross-validation. The initial input vector consists of all features extracted, $\mathcal{F}$.

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

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

4. Evaluate all feature subsets obtained using the preference learning approach presented in Section 4.3.2.

Genetic feature selection (GFS) implements a generational genetic algorithm to search for the set of features that yields the most accurate preference predictor for the investigated affective state [16]. According to the GFS mechanism, the whole set of input features are encoded as a bit string chromosome,

$$c = (g_1, g_2, ..., g_{N_\mathcal{F}}) \quad (1)$$

where,

$$g_i = \begin{cases} 1 & \text{if feature } i \text{ is included} \\ 0 & \text{if feature } i \text{ is not included} \end{cases} \quad (2)$$

and $N_{\mathcal{F}}$ is the total number of features in the dataset.

A population of N_c chromosomes is initialized with all bits set to zero but one selected randomly; i.e. the first generation consists of sets of one randomly selected feature. The reason for initializing chromosomes with only one feature is because we desire to obtain minimal feature subsets which yield high performing ANN predictors of reported affect. Then, at each generation:

1. All chromosomes of the population are evaluated. For that purpose a preference model is trained via neuroevolutionary preference learning for each chromosome and its performance is assessed via 3-fold cross validation. The fitness of each chromosome is the average 3-fold classification of the ANN trained on the feature set presented by the chromosome.

2. An elitism selection method chooses the best N_p individuals to be the parents of the next generation.

3. Pairs of parents are selected using a rank selection method that ranks the parents by their fitness and then selects two of them with a probability proportional to $\frac{1}{2+n}$ where n is the position in the ranking.

4. A total of $N_c - N_p$ offspring are reproduced via uniform crossover with probability p_c. If crossover is not applied, the most fit parent of the two is cloned to generate an offspring.

5. For each offspring, mutation occurs at each gene with probability p_m. The mutation scheme used flips the value of the selected gene which, in turn, suggests that the corresponding feature is either added (1) or removed (0) from the feature set. Finally, all offspring are inserted to the population.

The algorithm terminates after G_{max} generations are completed and the set of features corresponding to the highest performing preference predictor found across all generations is chosen. It is noteworthy that parent chromosomes are cloned to the new generation but their performance is reevaluated, i.e. a new ANN is trained on that feature set. Therefore, due to the non-deterministic nature of the neuroevolution, the fitness function of some individuals may fluctuate significantly from one generation to the next.

4.3.2 Preference Learning

After feature selection, the next stage of player modeling includes training a neural network using the input gameplay and controllable feature data acquired in user testing to provide a mapping to the output features that are the subjective preference ratings acquired in user testing. Preference learning is a machine learning research area that aids in the process of exploiting a set of specific features of an individual in an attempt to predict her preferences, and is the approach that will be employed here. It should be mentioned that the form of preference learning that is being implemented in this study is that of instance preference learning as opposed to label preference learning, since the goal here is to determine the relation between game levels and player preferences rather than predicting game levels that are more likely to be preferred among a set of alternatives [12].

An instance preference learning (IPL) problem consists of a set of $\mathcal{X}$ instances which are associated with a total or partial order relation and a set of $\mathcal{D}$ pairwise relations on them. The instances in our case are attribute-value features that consist of gameplay and controllable features that are generated during a gameplay session. The pairwise relations are the subjective data that is collected after a user plays a pair of game levels and rates their experience across the affective response dimensions that were specified in Section 4.2.2. The goal of the instance preference learning algorithm is to learn a relation between the feature-value instances and the pairwise preferences generated in the gameplay sessions. In more detail, IPL learns a preference relation $\mathcal{P}_x \subseteq \mathcal{X} \times \mathcal{X}$ for any instance $x \in \mathcal{X}$ given the pairwise preferences $\mathcal{D} = \{v_k \succ u_k | k = 1, \ldots, m\}$.

Overall, for the affective modeling problem under investigation the mechanism attempts to approximate a function that predicts whether $u_1 \succ u_2$ holds, where u_1, u_2 represent particular instances of data features, for any $u_1, u_2 \in \mathcal{X} \subset \mathcal{R}_d$ given [32]:

- a set of d features (e.g. features extracted from user-system interaction data or physiology) characterizing an interaction session (or interaction instance).
- a set of n training instances $\mathcal{X} = \{x_i | i = 1, \ldots, n\}$ comprising vectors of the measured values of those features for the (various) system variants.
- a set of m pairwise preferences $\mathcal{D} = \{v_k \succ u_k | k = 1, \ldots, m\}$ in which subjects report which of two variants they preferred ($u_k, v_k \in \mathcal{X}$ are the feature-value vectors of the specific feature instances concerned).

Given the nature of the preference learning problem, there are no specific target output categories for the learning problem, so gradient based learning methods such as back-propagation are inapplicable. For this reason, a genetic algorithm (GA) approach is applied for evolving the neural network. A population of $N_{population}$ networks is initialized randomly. Initial real values that lie within $[-5, 5]$ are picked for their connection weights randomly from a uniform distribution.

The learning algorithm proceeds as follows:

1. Each ANN in the population gets input two feature vectors, $d_{j,A}$ and $d_{j,B}$, which are the feature vectors generated by a player in pair j for game variants A and B and are normalized to have value in the range of $[0, 1]$. In each case the network returns two values, $e_{j,A}$ and $e_{j,B}$, representing the corresponding level of computed emotion in each variant.

2. The two computed outputs for each are fed into a logistic sigmoidal function that measures the difference between the reported affective preferences of the user and the corresponding ANN output values e. The logistic function is defined as:

$$g(e, \epsilon) = \frac{1}{1 + e^{-\epsilon e(f_k)}} \tag{3}$$

   where $e = e_{j,A} - e_{j,B}$ is the difference of the ANN output values for the pair of games, A and B for pair j, and $\epsilon = 30$ if $A \succ B$ and $A \prec B$. The values for ϵ were chosen from previous successes in preference learning applications [33] and [35].

3. Each member i of the population is evaluated with the fitness function defined as:

$$f_i = \sum_{j=1}^{N_{pairs}} g(d_j, \epsilon) \tag{4}$$

4. Roulette wheel selection is used to determine which chromosomes will be selected for crossover. The roulette wheel selection formula is:

$$p_i = \frac{f_i}{\sum_{I=1}^{N} f_i} \tag{5}$$

5. Selected parents clone an equal number of offspring so that the total population reaches N members or reproduce offspring by crossover. The Montana/Davis crossover operator is applied with probability 0.05 [17] [34].

6. Gaussian mutation occurs in each gene (connection weight) of each offspring's genome with a small probability $p_m = \frac{1}{n}$, where n is the number of genes.

The algorithm terminates after either a good enough solution is found or after a large number of iterations has completed (e.g. 10000).

4.3.3 Topology Optimization

Based on the data generated above and the features selected by the highest performing feature selection algorithm a multi layer perceptron (MLP) neural network will be constructed using the same neuroevolutionary approach as above. The difference between the SLP approach above and the MLP approach here is that the network will be optimized for 2 additional layers of hidden neurons with an estimate of 30 and 10 neurons in each layer. These numbers were acquired from a similar approach in [20] but their application used different features, feature selection methods, and number of output features. In the study several different MLP topologies will be trained and compared for tuning the topology with the highest accuracy.

4.4 Evaluating Game Content

Once topologically optimized player models have been generated, potential generated content for additional, adaptively created levels must be evaluated for their ability to produce better levels. The content is evaluated on it's usefulness/fitness in producing the desired affective response in the player. As outlined in [31], there are several different ways of evaluating the content that can be generated using different types of evaluations functions. The evaluation functions fall into one of several categories, based on their structure. The categories are direct, simulation-based, and interaction-based evaluation functions. These functions and the details of the approach to be used in this study are outlined in this section.

4.4.1 Direct Evaluation Functions

Direct evaluation functions involve the extraction of features from generated content, and a direct mapping from the extracted features to some fitness value for the content. Such features, for example can be the number of enemies in a game level, the layout of a game level, or many other properties of game content.

There are also two different types of functions within the classification of being a direct evaluation function. The two types are:

- Theory driven: evaluation based on theories of interaction, designer intuition, qualitative emotion theory, and other methods of deriving a link between the player experience model and the content quality.
- Data driven: evaluation based on the collection of data on the effects of various pieces of content, either with questionnaires or physiological measurements, and then automatically tuning the mapping from content to player experience and then to evaluation function.

The implementation and application of direct evaluation functions is the most studied area of player modeling and search-based PCG methods, and is the approach that will be most closely followed in this proposed work.

4.4.2 Simulation-based Evaluation Functions

Simulation based evaluation functions use artificial agents to play through parts of game levels where content evaluation should take place. The artificial agent can be hand-coded to act as a real player would, or generated through learned player behavior to simulate the actions of a player playing the game level. The simulated player behavior is then evaluated using the previously generated player models to

There are also two different types of simulation-based evaluation functions:

- **Static**: evaluation based on an agent that acts the same in any any situation while simulating play of the game.
- **Dynamic**: evaluation based on an agent that can act differently over the course of the game simulation. Dynamic evaluation approaches utilize some learning methods and the quality of the evaluation can be how well the agent is able to learn about it's environment.

Simulation-based evaluation approaches can be limited in several ways. They operate under the assumption that the agent plays like a human would, which has been proven to be difficult to actually accomplish in several user studies. Simulation-based approaches are also generally more computationally expensive than other approaches, and current methods can not be used for online content generation. Additionally, simulation-based approaches can only be used with models generated with gameplay-based data.

4.4.3 Interaction-based Evaluation Functions

Interaction-based evaluation functions use data from player preferences and information about content interaction during gameplay to update that quality score of the content used in the evaluation function. This can occur in one of two ways:

- **Explicit evaluation**: content quality is evaluated through questionnaires or verbal input data.
- **Implicit evaluation**: content quality is evaluated through interaction evaluation, such as how long a player interacts with an object, how much time a player spent exploring a particular room, or with physiological data such as how long the player's gaze rested on a certain type of object.

4.4.4 Dungeon Content Evaluation Function

The evaluation method proposed here will be a purely data-based direct evaluation function. The evaluation function will be composed of features generated from the reduced feature sets in the player models, and the controllable features of the level, mapping player type and game layout to reported affective preference. Essentially the fitness is the direct result of the neural network output for the player input data.

In addition, the interactive portion of the content evaluation will occur as an intermediate step in the running of the game, soliciting explicit preference ratings from the players after each level is completed, adding to the amount of data that the model will have for future training iterations. This will not contribute to the generation of content in real-time, but is worthy to note since the overall player model will be adapting to new players over time.

A diagram depicting an overview of the different content evaluation methods and the general approach selected in this proposed work is shown in Figure 19.

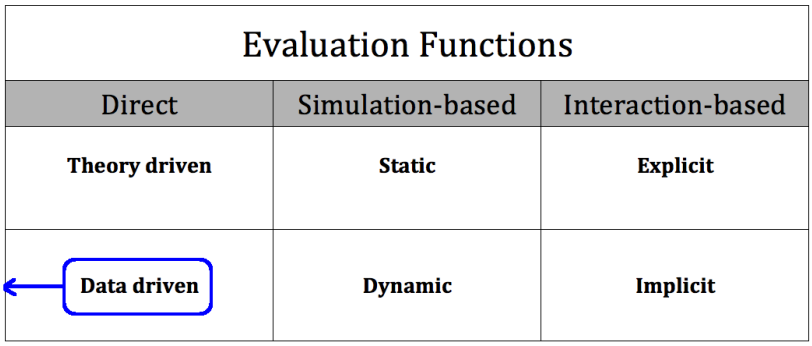


Figure 19: Evaluation function taxonomy and the approach to be used in this study circled in blue.

4.5 Generating Game Content

Once the player models have been created, and their associated evaluation functions constructed, content can then be generated during an instance of a running game. In order to generate the best content for the player, the space of possible content arrangements must be searched over for the level that optimizes the affective response for the player. The space of possible game content arrangements is determined by the dimensionality of the controllable features for the level and the method of search will be an evolutionary algorithm (EA) in the form of a genetic algorithm.

4.5.1 Content Representation

There are lots of different ways that content can be represented in games. In order to reduce the dimensionality of the search space, some level of abstraction must be performed in determining how to represent the game content. In their taxonomy paper, Togelius, et al., describe this process as indirect modeling of game content [31]. A diagram depicting the taxonomy of content representation and generation approaches is shown in Figure 20.

Indirect modeling is the process of describing game components in abstract ways. Considering the dungeon environment, a level could be described as a feature list with parameters for every

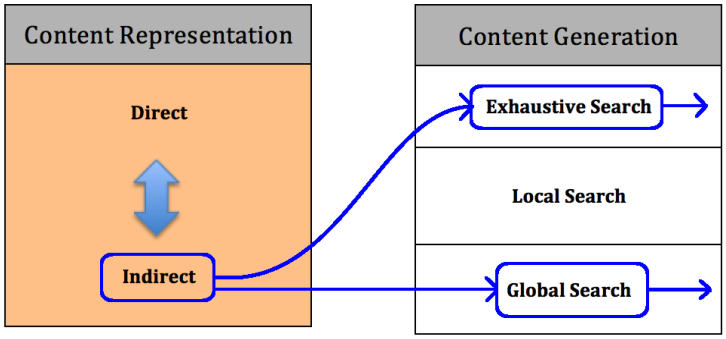


Figure 20: Content representation and generation taxonomy and the approach to be used in this study circled in blue.

possible location of room, wall, corridor, enemy, and item type. This representation would yield the most comprehensive genotype that could be used in the genetic algorithm for searching the space of content, but would also result in a search space that is far too large to be searched over in any reasonable amount of time.

Given our dungeon game environment, the level geometry generator that gives a great deal of flexibility in generating level environments by only adjusting a few level parameters. Also to be integrated are parameters that can be adjusted for enemy difficulty, item frequency, and all of the other controllable features of the game levels, but once they are integrated the result will be a relatively low dimension feature space that is capable of representing many levels.

The controllable features for the level also serve as the parameters for the space of levels that can be generated. The features, and their associated ranges of values that they can take on is shown in Table 9.

Feature	Range	Step Size	Possible Values
Level length	$[4, 13]$	1	10
Topology variety	$[0, 14]$	1	15
Enemy variety	$[1, 5]$	1	5
Enemy frequency	$[1, 8]$	1	8
Bosses	$[0, 2]$	1	3
Trap frequency	$[0.2, 2]$	0.2	10
Chest frequency	$[0.2, 2]$	0.2	10
Health frequency	$[0.2, 2]$	0.2	10

Table 9: The controllable features and the ranges that they can take on.

4.5.2 Searching and Generating Content

There are different options for how to complete the search, mostly dependent on the dimensionality of the feature space. Given the parameters and the ranges of the values there are 18,000,000 distinct levels that can be generated. With this search space, it is possible that exhaustive search will be able to produce a level that optimizes the player's experience for the next play session relatively

quickly, but in the case that it is not, a genetic search approach will also be explored for finding the best level during a fixed period of time. As such, the game parameters will be able to be adjusted in an online fashion during the execution of a level, as long as a sufficient period of time is given to delay the play of the game.

The exhaustive search will attempt every possible combination of level parameters, evaluating each one and choosing the set of parameters that are best. Parameter combinations that are not feasible will not be considered, i.e., if a player has already explored 4 rooms then only new level parameters that have 4 or more rooms will be evaluated for fitness.

The genetic search algorithm will proceed in a similar fashion to the approach above for tuning the ANN that is the player model. A population of levels will be initialized with some random values and the levels with the highest fitness will be combined and mutated in the next generation. The algorithm will terminate after a fixed number of iterations or until a set of parameters has been found that doesn't significantly change over a preset number of iterations. Again, only level parameters that are feasible given the previously explored parts of the level will be considered for in the next generation.

The main contribution of this work is that ED-PCG has never (at the time of this writing) been used to generate game content during the execution of a game level, and certainly not at the scale that is being attempted in this work. To accomplish this there are several modifications that will have to occur in the generation algorithm that is integrated in the Pro-D level generator. As it stands, the base dungeon generation algorithm creates an entire new map each time the map generator is triggered, based on the parameters that are set in the map generation UI. In order to procedurally generate level content that is better suited to the playing style of the individual, it must be possible to generate parts of levels that haven't been played yet but the parts of the level that have been explored already must remain the same.

This change to the generation algorithm also requires that the data that is acquired must be aggregated over time during a gameplay session, and used to incrementally improve the quality of the level as it is being played. To make the transition between the different level parameters, a cutscene will be displayed at key points during the play of a level where play will pause for a cut scene and the Goblin Shaman will enter to tell the player that he is changing the dungeon to make his escape more difficult. A diagram depicting the flow of a level using this methodology is depicted in Figure 21.

4.6 Data Analysis

To evaluate the effectiveness of the ED-PCG approach, the difference between the optimized values of each new set of level parameters and the actual game parameters will be computed at each content generation point during the play of the level. The effectiveness of the ED-PCG approach can partially be measured by whether the final optimization requires more or less adaptation than the previous content optimizations during a level. If the previous optimizations were effective, then the final optimization should ideally require no change to the level parameters in order to provide a better player experience.

In addition to this data based approach of evaluating the effectiveness of the ED-PCG methods a further set of user tests will be performed. The final user-testing process will include a fixed response survey that will be used for explicit subjective responses as to the effectiveness of the ED-PCG methods. Similarly to the user tests that were conducted in the data gathering phase of

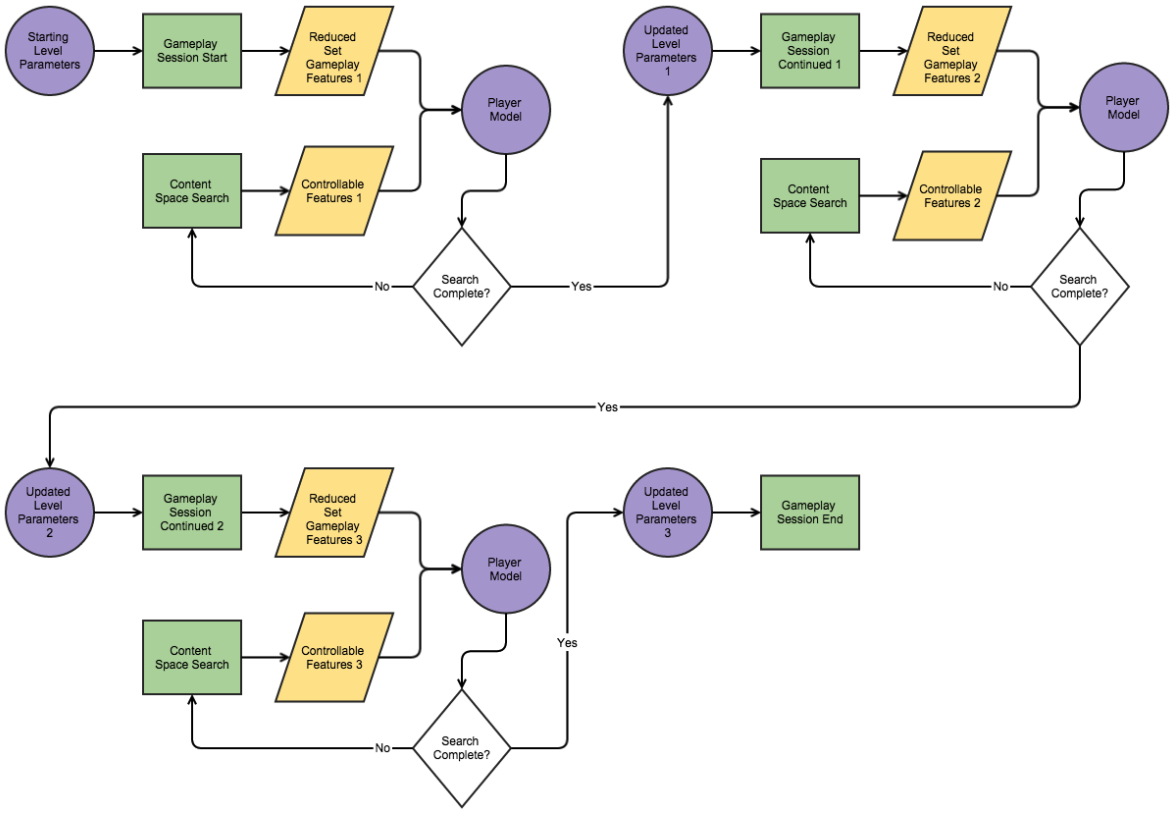


Figure 21: Content search approach depicted in the flow of a level.

the project, users will be asked how they felt about the level across the same dimensions mentioned in Table 7 but without having to compare 2 different level play-throughs. In this survey, the users will be asked to compare their feelings between the first, second, and third parts of the game level (separated by Goblin Shaman visits and content optimizations). As above, the users will be surveyed in a 4-AFC protocol but in this case A and B represent the different parts of the game level and E is one of the emotional states defined in Table 7.

This additional survey will provide more data for the models to be trained on, ideally further improving the accuracy of the player models and the performance of the content generation over time. The models will not be updated in real-time initially, since the training of the neural networks is a considerably longer process than the content search, but will be updated offline, over time after the initial models are created and the content generation algorithms implemented. The updated results will require a different learning process than preference learning, since the subjective data acquired will not be in the form that is feasible for preference learning, but since this is not a requirement for the system, and could take a lot longer than expected to achieve noticeable results, it will not be considered as a necessary goal for the data analysis.

Analysis of the neural network training accuracy and the content generation statistics will also be thoroughly explored. In particular the neural network will be assessed on:

- **Accuracy:** the theoretical accuracy of the model, after the initial

- **Feature subset**: the results of the feature selection approaches, a subset of the entire feature set that results in the highest prediction accuracy.
- **Individual feature contribution**: also a result of the feature selection techniques, the individual contribution of gameplay and controllable features toward subjective preference accuracy will be analyzed.
- **Training time**: the time it takes to train the ANN to it's highest accuracy will be reported for both the SLP and MLP topologies.

The content generation search will be evaluated assessed on:

- **Accuracy**: how close to optimal for the affective dimensions that the content search techniques can achieve.
- **Time to converge**: the amount of time it takes the genetic and exhaustive algorithms to converge to a solution.

The final evaluation of the system will be the effectiveness of the ED-PCG methods measured in comparison with the players' final subjective responses as discussed above.

References Cited

[1] 3dfoin. Manticore. http://3dfoin.com/manticore.html, 2014.

[2] 3DRT. Troll golem. http://3drt.com/store/characters/fantasy-characters/troll-golem.html, 2014.

[3] Cameron Alston and Arnav Jhala. Automating camera control in games using gaze. In *Workshops at the Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[4] Entertainment Software Association. Essential facts about the computer and video game industry. http://www.theesa.com/facts/pdfs/ESA_EF_2014.pdf, 2014.

[5] badfluid. 5 animated medieval traps. https://www.assetstore.unity3d.com/en/#!/content/6724, 2014.

[6] Regina Bernhaupt. *Evaluating User Experience in Games: Concepts and Methods*. Springer Publishing Company, Incorporated, 1st edition, 2010.

[7] BitGem. Low poly dungeon entrourage set. http://shop.bitgem3d.com/collections/low-poly-3d-props/products/low-poly-dungeon-entourage-set, 2014.

[8] CaptainKraft. Create a procedurally generated dungeon cave system. http://gamedevelopment.tutsplus.com/tutorials/create-a-procedurally-generated-dungeon-cave-system--gamedev-10099, 2014.

[9] Stuart K. Card, Allen Newell, and Thomas P. Moran. *The Psychology of Human-Computer Interaction*. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 1983.

[10] Ben Cowley, Darryl Charles, Michaela Black, and Ray Hickey. Toward an understanding of flow in video games. *Comput. Entertain.*, 6(2):20:1–20:27, July 2008.

[11] Dogzer. Adventurer animation showcase. https://dl.dropboxusercontent.com/u/11214337/adventurer/armor/WebPlayer.html, 2014.

[12] Johannes Fürnkranz and Eyke Hüllermeier. *Preference learning*. Springer, 2010.

[13] George Alex Koulieris, George Drettakis, Douglas Cunningham, and Katerina Mania. An automated high level saliency predictor for smart game balancing. *ACM Transactions on Applied Perception*, 2014.

[14] A. Liapis, G.N. Yannakakis, and J. Togelius. Adapting models of visual aesthetics for personalized content creation. *Computational Intelligence and AI in Games, IEEE Transactions on*, 4(3):213–228, Sept 2012.

[15] Polygon Maker. Fantasy horde - enemies. http://www.polygonmaker.com/Products/Products_enemies.html, 2014.

[16] Héctor Pérez 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*, AFFINE '10, pages 15–20, New York, NY, USA, 2010. ACM.

[17] David J Montana and Lawrence Davis. Training feedforward neural networks using genetic algorithms. In *IJCAI*, volume 89, pages 762–767, 1989.

[18] Lennart Nacke. Towards a framework of player experience research. In *In Proceedings of EPEX?11 at FDG 2011*, 2011.

[19] Mr. Necturus. Goblin shaman. `http://mr-necturus.com/shop/goblin_shaman_unity`, 2014.

[20] C. Pedersen, J. Togelius, and G.N. Yannakakis. Modeling player experience for content creation. *Computational Intelligence and AI in Games, IEEE Transactions on*, 2(1):54–67, March 2010.

[21] Frank Rosenblatt. The perceptron–a perceiving and recognizing automaton. Technical Report 85-460-1, Cornell Aeronautical Laboratory, 1957.

[22] Adam M. Smith, Chris Lewis, Kenneth Hullet, and Anne Sullivan. An inclusive view of player modeling. In *Proceedings of the 6th International Conference on Foundations of Digital Games*, FDG '11, pages 301–303, New York, NY, USA, 2011. ACM.

[23] Gray Lake Studios. Prod-d web player. `http://graylakestudios.com/graylakestudios/uas/prod/prod_webplayer_editor`, 2014.

[24] J. Togelius, R. De Nardi, and S.M. Lucas. Towards automatic personalised content creation for racing games. In *Computational Intelligence and Games, 2007. CIG 2007. IEEE Symposium on*, pages 252–259, April 2007.

[25] J. Togelius, G.N. Yannakakis, K.O. Stanley, and C. Browne. Search-based procedural content generation: A taxonomy and survey. *Computational Intelligence and AI in Games, IEEE Transactions on*, 3(3):172–186, Sept 2011.

[26] 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. Togelius is working at IDSIA on SNF grant 21-113364 to J. Schmidhuber.*

[27] Mélodie Vidal, Ken Pfeuffer, Andreas Bulling, and Hans W. Gellersen. Pursuits: Eye-based interaction with moving targets. In *CHI '13 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '13, pages 3147–3150, New York, NY, USA, 2013. ACM.

[28] Georgios N. Yannakakis and John Hallam. Entertainment modeling through physiology in physical play. *International Journal of Human-Computer Studies*, 66(10):741 – 755, 2008.

[29] 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, October 2010.

[30] Georgios N. Yannakakis, Pieter Spronck, Daniele Loiacono, and Elisabeth Andre. Player modeling. In *Artificial and Computational Intelligence in Games*. 2013.

[31] Georgios N. Yannakakis and Julian Togelius. Experience-driven procedural content generation, 2011.

[32] 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, Sept 2009.

[33] G.N. Yannakakis and J. Hallam. Game and player feature selection for entertainment capture. In *Computational Intelligence and Games, 2007. CIG 2007. IEEE Symposium on*, pages 244–251, April 2007.

[34] G.N. Yannakakis, H.H. Lund, and J. Hallam. Modeling children's entertainment in the playware playground. In *Computational Intelligence and Games, 2006 IEEE Symposium on*, pages 134–141, May 2006.

[35] G.N. Yannakakis, M. Maragoudakis, and J. Hallam. Preference learning for cognitive modeling: A case study on entertainment preferences. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 39(6):1165–1175, Nov 2009.

Project Logistics

5 Logistics

5.1 Schedule

5.1.1 Winter Quarter 2015

- Experimental environment development
  1. Acquire assets from Unity Asset Store and set up project repository (1 day).
  2. Integrate acquired assets to create a single project structure (2 weeks).
  3. Adding objective and gameplay telemetry into game environment (3 weeks).
  4. Create hand crafted levels that target specific affective responses from literature (4 weeks).
- User study
  1. Design user study questionnaire (1 day).
  2. Advertise user study (1 day).
- Paper submission
  - CHI 2015: Doctoral consortium paper
  - FDG 2015: Doctoral consortium paper

5.1.2 Spring Quarter 2015

- User study
  1. Implement online testing functionality (1 week).
  2. Open up online environment (1 week).
  3. Acquire data online (∼8 weeks).
  4. Acquire data in person with physiological data (∼4 weeks).
- Player modeling
  1. Implement and test the machine learning algorithms described in Section 4.3 (2 weeks).
- Paper submission
  - AIIDE 2015: Player modeling integrating different data streams for ED-PCG.
  - CIG 2015: Player modeling integrating different data streams for ED-PCG.

5.1.3 Summer Quarter 2015

- User study cont.
    1. Continued data acquisition online (∼6 weeks).
- Player modeling cont.
    1. Continued data mining experiments for construction of content evaluation models (∼ weeks).
    2. Run feature selection (2 weeks).

5.1.4 Fall Quarter 2015

- Content generation
    1. Integration of content generation functionality to dungeon game (∼8 weeks).
    2. Setting up final user study (∼1 week).
- Paper submission
    - AIIDE 2016: Real-time ED-PCG.

5.1.5 Winter Quarter 2016

- User study
    1. Final user study
- Paper submission
    - ACII 2016: Real-time ED-PCG.

5.1.6 Spring Quarter 2016

- Thesis writing
- Paper submission
    - FDG 2016: Real-time ED-PCG.

5.1.7 Summer Quarter 2016

- Thesis writing/completion

5.2 Other Personnel

5.2.1 Undergraduate Programmers

To help with the initial construction and integration of the game environment, undergraduate programmers could be employed as undergraduate research assistants earning academic credits. This could significantly alleviate the amount of time and work necessary for getting the test environment off the ground in terms of developing the infrastructure for the research experiments.

5.2.2 User testers

There will be 2 different types of testers used over the course. The first kinds of testers will take part in the user study for generating the data to be used in the player models. The second type will be used for testing the game and ED-PCG functionality after the models have been learned and implemented.

5.3 Costs

5.3.1 Unity Development Tools

The costs of the models and development tools that are proposed in the paper are summarized below.

Item	Description	Source	Cost
Pro-D Total	Procedural dungeon generation tool	Gray Lake Studios	$75
Adventurer Character	High poly hero model, rigged and animated with sword and shield	Dogzer	$15
Fantasy Horde - Enemies	Multiple rigged and animated enemy models with different armor and weapons	Polygon Maker	$95
Troll Golem	Animated large enemy model	3DRT	$35
Manticore	Large dragon enemy	3dFoin	$15
Dungeon Props Pack	100+ items and objects for use in decorating a dungeon environment	BITGEM	$35
		Total	$270

5.3.2 Hardware

No additional hardware should be required to implement the game functionality.

5.3.3 User Testing

In order to incentivize users to participate in the data collection and subsequent testing experiments, it would be advantageous to provide opportunities for the users involved in the testing study to eat food or win prizes. Possible incentives are listed below.

Item	Description	Quantity	Cost
Xbox One	Raffle grand prize	1	$349
Amazon gift card	Raffle second prize	4	$25
Donuts	Participant food	6	$15
		Total	$539

Biographical Sketch: Cameron Alston

Maximum of 2 pages

Biographical Sketch

Professional Preparation

Undergraduate Institution(s)	Major	Degree	Year
Sacramento City College	Computer science	N/A	2007
University of California, Davis	Computer science	Bachelors	2009

Graduate Institution(s)	Major	Degree	Year
University of Southern California	Computer science game development	Masters	2011
University of California, Santa Cruz	Computer science	PhD	2016 (exp.)

Year(s)	Position, Department, Institution
2013, 2014	Lead Instructor, Pre-Collegiate Studies, Stanford University
2012	Lead Instructor, Digital Media Academy

Publications

Author Publications Most Closely Related to the Proposed Project

1. Cameron Alston and Arnav Jhala: Automating Camera Control in Games Using Gaze, Workshops at the Twenty-Eighth AAAI Conference on Artificial Intelligence, 2014.

Collaborators & Other Affiliations

Thesis Advisor and Postgraduate-Scholar Sponsor: Jim Whitehead, UCSC