CHAPTER 1

TOWARDS THE ADAPTIVE GENERATION OF BESPOKE GAME CONTENT

Cameron Browne, Simon Colton, Michael Cook, Jeremy Gow and Robin Baumgarten

Computational Creativity Group, Department of Computing, Imperial College, London

In this chapter, we explore methods for automatically generating game content — and games themselves — adapted to individual players, in order to improve their playing experience or achieve a desired effect. This goes beyond notions of mere replayability, and involves modelling player needs to maximise their enjoyment, involvement and interest in the game being played. We identify three main aspects of this process: *Generation* of new content and rule sets; *Measurement* of this content and the player; *Adaptation* of the game to change player experience. This process forms a feedback loop of constant refinement, as games are continually improved while being played. Framed within this methodology, we present an overview of our recent and ongoing research in this area. This is illustrated by a number of case studies that demonstrate these ideas in action over a variety of game types, including: 3D action games, arcade games, platformers, board games, puzzles and open world games. We draw together some of the lessons learned from these projects to comment on the difficulties, the benefits and the potential for personalised gaming via adaptive game design.

1.1 Introduction

Personalisation of games for individual players is seen as a big future marketing factor for games, and is currently a major driving force for improved game design, which will ultimately lead to better games and happier, more engaged and entertained customers. Within this scope, there is a particular nirvana wherein games automatically adapt before, during and after being played to take into account the style, experience and personality of each player. Of course, games have always had a simplistic adaptive element, whereby stronger players progress to play more difficult levels, to keep them interested. However, this type of adaptation only takes into account their skill level at that particular game, and ignores other information such as their likes, dislikes, temperament, current mood and overall ability. Such information can in principle be gathered through gameplay, sensors, surveys and other routes, and will be used in adaptive gaming technologies of the future to generate bespoke games that truly change to fit an individual player, greatly enhancing their playing experience.

The automatic adaptation of games to players is also a major force for applied Artificial Intelligence (AI) research. In particular, as a research group, in addition to the long term goal of improved games, we are also interested in studying games from the perspective of the subfield of AI known as Computational Creativity research [17]. In this area, we study how to engineer software which can take on some of the *creative responsibility* in arts and science projects. In this context, games, and video games in particular, can be seen as a 'killer domain' for creativity research. This is largely because generating a game requires the generation of all the types of artefact we usually produce individually, including audio (sound effects, music); graphics (characters, backdrops); text (dialogue, plotlines) and concepts (puzzles, rulesets, interaction schema, game mechanics). However, there are many other advantages to working with games as a medium within which to study Computational Creativity. These include: (a) the fact that the output is entirely digital and the audiences are entirely online, hence requiring no exhibitions, concerts, readings, publications or demonstrations in order to get culturally relevant feedback (b) a general acceptance of automated processes as being valuable, which isn't always true in more traditional artistic circles (c) a requirement to model and ultimately alter both positive and negative emotions (d) an interesting balance between the entertainment value and the intellectual value of games and (e) explicit requirements to incorporate user engagement and interaction in the generated artefacts.

As a group of Computational Creativity researchers and avid gameplayers, over the last five years, we have eagerly investigated the potential for automating processes related to game design, with the specific long-term goal of adaptive game generation in mind. We see adaptive systems in games — also known as AI directors or gamemasters — as a form of procedural content generation which aims to enhance the players' gaming experience by delivering personalised game content. When thinking about such adaptive systems, it helps to consider various aspects such as the type of *player data*; the types of decisions to be made about game content (*the content output space*); how the latter is computed from the former (*the*

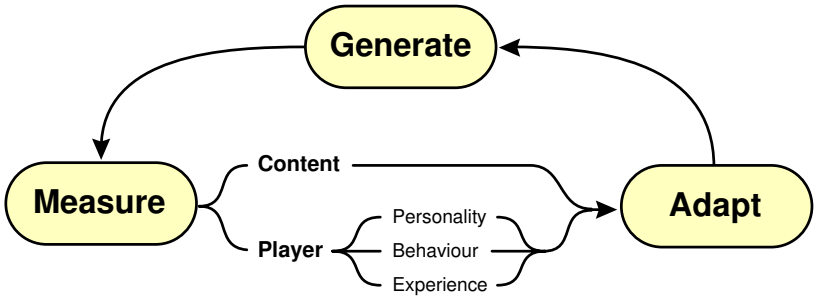

Figure 1.1 An overview of the adaptive game generation process.

adaptive mechanism); and the desired effect on player experience (*the adaptation requirements*). We have studied the potential for adaptive games with a shotgun approach, i.e., numerous projects involving games of various genres, which address all the above aspects. We present here an overview of some of these projects, in order to highlight the lessons learned, difficulties encountered and huge potential for adaptive game technology to both help produce next generation games and to stimulate research in Computational Creativity.

In section 1.2, we describe an overall methodology within which content generation for adaptive games can take place. This centres around a cycle of generation, measurement and adaptation, and we expand each of these aspects further. With respect to generation, we place this in a context of search-based procedural content generation, and focus on two types of evolutionary search. With respect to measurement, we split this into measuring the game, measuring the player and measuring the adaptations. Finally, we place adaptation into a broader context of improving player experience, and cast it as a machine learning problem. In section 1.3, we describe various projects where we have studied aspects related to automating adaptive game design, with respect to the methodology given in section 1.2. These projects cover different genres of games with which we have experimented, including 3D action games, platformers, arcade games, board games, puzzles, and open world games. In the final section of the chapter, we take an overview of these projects and draw conclusions about the prospects for personalised gaming through adaptive game design.

Note that it is beyond the scope of this chapter to cover all the work done in the area of adaptive content generation, and we only present background material which is direcly relevant to the projects we describe. Each of those projects is covered by various of our research papers which we cite in the chapter, and which can be referenced for further literature reviews.

1.2 Methodology

Given the need for the adaptive generation of bespoke game content, this section describes *how* this can be achieved for digital games. We focus on the processes

that we have used for projects ourselves, but which have broader application to other domains. In each case, the process involves three fundamental steps, summarised below:

1. *Generation* of new content and rule sets.
2. *Measurement* of the generated content and target players during adaptive generation, or as part of system design and evaluation.
3. *Adaptation* with the aim of changing a target player's gaming experience.

These steps are summarised diagrammatically in Figure 1.1, where the arrows indicate the order of operation. In the following subsections, we consider each of these steps in detail.

1.2.1 Generation

The first step in the cyclic adaptive process is the generation of novel game content and game rules. This may be achieved through fully automated means, although a significant amount of our research also investigates the use of the computer as a *creative collaborator*, that assists the designer by taking on some creative responsibilities. In this section, we describe the generational methods most commonly used in our work.

1.2.1.1 Procedural Content Generation The exponential growth of digital games in recent years means that there are now hundreds of millions of people playing games every day, wanting new and interesting content [29]. However, the related production costs and requirements for specialised manual labour to develop content to satisfy this demand have also increased exponentially, and the industry is now facing serious scalability issues. Games are becoming larger and more complex, with virtual worlds that are open, massive, and ongoing, which puts impossible demands on designers and artists alike and creates a *content creation bottleneck*.

Procedural content generation (PCG) — the automatic creation of content through algorithmic means — offers a potential solution to this shortfall between consumer need and industry output, and is becoming an increasingly important field of research for digital game design, for both the artistic content of games and for game play itself. *Content* in the context of digital games may refer to any of the following:

- *Rules* that govern the game-play.
- *Challenges* that define initial states posed to players.
- *Resources* that define the game's look, theme, feel, and so on.

PCG is a difficult task for creative domains such as game design, as the automatically generated content must satisfy the constraints of the designers and artists, as well as the (often poorly defined) needs of the end users. However, it offers the

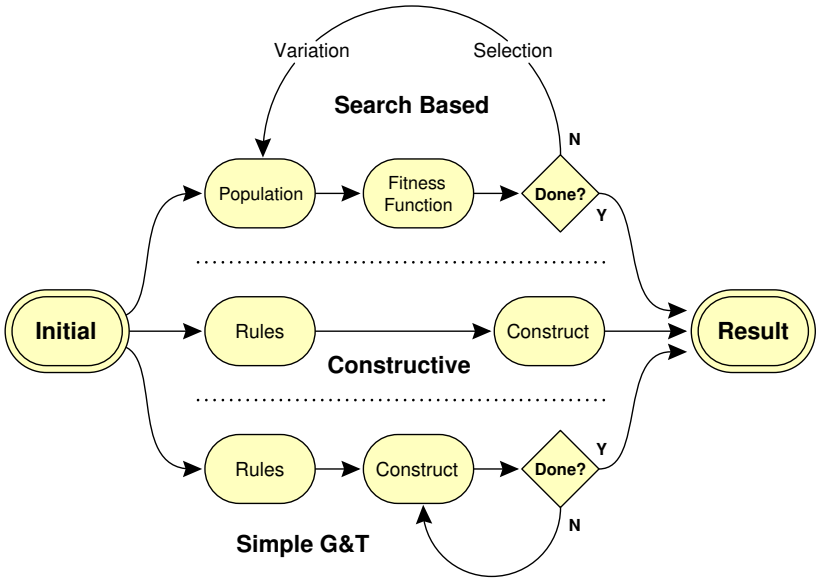

Figure 1.2 Main forms of PCG (from [58]).

promise of handing at least some of the creative responsibility to the computer, and we are now seeing an increasing amount of procedurally generated content in commercially released games.

Search-based procedural content generation (SB-PCG) is a particular type of PCG, in which a test function grades the generated content for fitness and guides the search for new content accordingly. As depicted in Figure 1.2, Togelius *et al.* [58] distinguish SB-PCG from other forms of PCG as follows:

1. *Search Based*: Content is iteratively generated, according to a fitness function that guides the search.

2. *Constructive*: Content is directly generated according to certain rules, with strict validation.

3. *Generate & Test*: Content is iteratively generated according to certain rules, and filtered for fitness.

SB-PCG is an ideal mechanism for adapting games and game content on-the-fly, in response to players' needs, as the system can learn and improve its output the more it is used. See [29] and [58] for further details on PCG and SB-PCG for games. The two main SB-PCG mechanisms we have used in our projects are evolution and co-evolution, as described below.

1.2.1.2 Evolution In traditional evolutionary systems, a population of possible solutions to a particular problem are evaluated for 'fitness' (some numerical value indicating how well they solve the problem) and recombined to produce hybrid solutions that hopefully inherit positive traits from the previous population. For a generative task such as those in procedural content generation, the task at hand is to produce a piece of content to meet certain quality or player-specific targets, and a solution is a piece of finished content that can be evaluated against those targets. The process of iterative evaluation and recombination is repeated until some stopping condition is met, which may involve measurements of the content produced.

Evolutionary algorithms are used in a wide variety of applications, including across the games industry. Evolution is particularly useful (a) where only general criteria for a solution can be stated — such as Paul Tozour's *City Conquest* (Intelligence Engine Design Systems, forthcoming), which used computational evolution to stress test the game for balance issues — or (b) where the space of possible solutions is so large that searching it using other methods is too difficult — for instance, the *Starcraft II* (Blizzard 2010) community was upset by a genetic algorithm that could optimise complex build orders and discovered exploits unknown to even the best human players.

Evolutionary algorithms tend to perform best when the fitness functions and the representation of a solution are relatively simple. For larger problems, where solutions may be very complex and fitness evaluations include many competing estimations of quality, evolutionary algorithms are harder to design optimally and take longer to produce good solutions. They also lack a guarantee of robustness: due to the random nature of the generation and recombination processes, even the best-designed evolutionary systems may produce bad or severely suboptimal solutions. This issue is one reason that evolutionary algorithms are more commonly used in pre-production to generate content that can be curated before inclusion. This problem can often be mitigated by building additional systems to perform quality checks or adjust evolutionary parameters, and many applications of content generation come with the expectation that the system may occasionally produce curios or eccentric output.

1.2.1.3 Co-Evolution Co-operative co-evolution (CCE) is a type of evolutionary algorithm that helps solve larger problems by decomposing them into smaller tasks that can be solved individually. In their paper proposing the algorithm [47], Potter and De Jong say that in order to evolve more complex structures, explicit notions of modularity need to be introduced in order to provide reasonable opportunities for complex solutions to evolve. These modules are called 'species', and are structured as self-contained evolutionary systems, with a population and a fitness function of their own.

The difference between a species and an ordinary evolutionary system is that a fitness function evaluates a member of its population in the context of the original design problem. That is, if we have a problem P decomposed into n evolutionary algorithms $P_1, \ldots, P_n$, in order to evaluate a candidate solution $s \in population(P_i)$, we gather the best known members of populations $P_1, \ldots, P_{i-1}, P_{i+1}, \ldots, P_n$ and

combine them with s to make a solution to the original problem P. The fitness function then evaluates s both on the quality of it as an individual solution, and the quality of its *co-operation* with the other $n - 1$ subproblems.

Co-operative co-evolution offers many benefits when building content creation tools. Each module can easily encapsulate a particular design task, such as level design, which helps conceptually separate the different elements of content creation. It is also easily amenable to mixed-initiative design, where a human contributes to the content generation process alongside an AI system. Because fitness functions react to the context provided to them by the other species, we can remove a CCE species and replace it with a static, human-generated piece of content, and the CCE system will design and adapt its other species to the content provided. For example, consider a puzzle-game designer that conceives of a rule set in one CCE species, and designs a set of levels with another CCE species. In normal execution, the evolution of the rule set will interact with the evolution of the level design, and over time the two will co-operate and complement each other. However, we might want to develop a particular kind of puzzle game. If we replace the rule set species with a static rule set that represents the mechanics we want to use, the level-designing species will design levels tailored to the human-designed rule set. This idea has enormous potential for improvisational game design tools, where software and designer play off one another's ideas.

Video game design represents a particularly complex problem, being comprised of many different components (such as levels, mechanics, artwork, narratives and music) all of which have different estimations for their fitness, and depend on each other for their definitions of quality; a 'good' level for a set of mechanics like those in *Pac-Man* (Namco 1980), is very different from a good level for a game such as *Doom* (id Software 1993). Standard evolutionary systems would need to take into account a vast array of quality estimations that would change while the evolution was still taking place, but CCE allows us to subdivide and specialise these design tasks to better deal with each individually. For a description of a system employing CCE for the procedural generation of content, see section 1.3.3.

1.2.2 Measurement

The second step in the adaptive process is the measurement of the generated of content. This involves:

1. Measuring the quality of the generated artefacts (game content and rules), according to specified criteria or desired aims. This can be achieved, for instance, through automated self-play.

2. Measuring how the player plays the game, taking external observations during play, and measuring other contextual factors such as personality or stored profiles.

Specific applications have often only used one approach, measuring the content *or* the player. But, in general, both can contribute to bespoke game design. Measurement can play three distinct roles in adaptive game generation systems as follows:

Adaptive measurement The system measures aspects of the generated content and target player, to deliver content adapted to that player.

Formative measurement The system designers test the quality of generated content and player's reaction to content and/or adaptation, in order to inform the design of the system. For example, this can include gathering player feedback to train a learning algorithm.

Summative measurement The working system is evaluated in terms of the quality of generated content and a player's reaction to content and/or adaptation.

Adaptive measurement is *autonomous*, carried out by the bespoke game design system, whereas formative and summative measurement are *human-guided*. In human-guided measurement, we are often able to exploit information not accessible to autonomous measurement, by the game during normal adaptive play. For instance, verbal feedback or physiological measurements. However, it is possible for such data to be measured autonomously during play by a sufficiently sophisticated system. In this subsections below, we consider some different approaches to measuring content and players.

1.2.2.1 Measuring the Game We understand the *quality* of a game to mean the potential for the game to *engage* players: the capacity of that game to interest players and to keep them in that state. Gauging the quality of a generated game or piece of game content can be difficult, as the notion of quality can depend on the context, and vary from player to player.

One approach is define *quality metrics* which provide a computational assessment of an aspect of game quality. Such metrics can be used to automatically guide the search during SB-PCG, but can also be useful during system design and evaluation. An alternative is evaluate game quality by *playtesting*, where explicit feedback is gathered from players. This is typically used as formative or summative measurement, but could also form part of the adaptive process, in systems which directly solicit players for feedback to guide adaptation.

Most research into game quality metrics has been done in the context of board games, where games of any significant depth tend to involve mechanisms and strategies that emerge during play, and which may not be obvious from their rules alone. For this reason, it is generally more reliable to measure board games for quality via the playing of games, rather than from the rules alone. This can be achieved by conducting series of self-play trials between artificial players. Many of the metrics can, in principle, be generalised to evaluate the quality of video games. AI players, called *bots*, can be used to automatically test generated video game content and gather metric data, although for more complex games, creating an AI-bot may be a very time-consuming task.

Browne [9] describes 57 aesthetic criteria for empirically measuring the quality of board games, mostly from trends observed during self-play trials. These include interpretations of the following four key features of abstract games, outlined by Thompson [57]:

Depth The capacity for a game to be played at different levels of skill, and to reward continued study.

Clarity The ease with which players can understand the rules and plan moves.

Drama The potential for players to recover from trailing positions, to eventually win the game.

Decisiveness The ease with which players can close the game out, once a winner is certain.

Other useful metrics include *uncertainty* [34], *balance* [31] and *game length* [1]. Game length has proven to be a particularly effective indicator of flawed games, as it quickly detects trivial games that end within a few moves, as well as strategically flawed games in which players can defend indefinitely with optimal play, and logically flawed games in which the goals simply can't be reached using the specified rules. Player testing with people is preferable if an appropriate quality metric is difficult or impractical to implement. In such scenarios, player testing can be more reliable, as the player is the end user that the game design process is ultimately trying to satisfy. Further details of measuring player experience are described in Section 1.2.2.4.

1.2.2.2 Autonomous Player Measurement To deliver bespoke game content, generation methods need to be based on data about a particular target player or players. This *player data* can be collected before play, to generate new content for the next game, or during play itself, in order to adapt upcoming content in the current game. Data on multiple players can be collected in order to generate common content for that group, either because they are playing a multiplayer game, or because they are being collectively targeted with the same content, e.g. as members of the same age group.

The player data most easily gathered by digital games are *gameplay logs*, which contains a record of in-game states and events, and player actions, from which summary player features may be computed, for instance, as in [7, 28, 52]. However, other forms of player data can be gathered, such as demographic data, motion, posture, physiological signals [41], visual appearance [3], retail activity, social media activity [49] and direct player feedback on experience. These data sources are not always available, but as mainstream gaming hardware develops (e.g. in the motion-aware Wii and Kinect consoles), and social media and gaming become more integrated, there is a growing commercial interest in exploiting these resources, as in [2]. Note that even when the available data is restricted, a content generation algorithm can be still be informed by other kinds of formative player measurement during design or training, e.g. feedback on player experience during testing [53, 63].

Player input data is often reduced to a set of categorical and/or scalar features. This provides a simplified input to the subsequent content generation stage, and allows the use of standard machine learning techniques. Given the range of possible player inputs, this feature data can measure any aspect of the player, his/her activity, and the context of play. As examples, we can take measurements of current situational intensity — as in *Left 4 Dead* (Valve 2008) [7] — weapon use [28], summary statistics for a individual combat [25], a single level [52] or an entire video game [23]. An alternative to scalar and categorical features is to use structured player data, such as paths, sequences, trees or graphs. To date, structured player data has been relatively unexplored, although there has been a growing interest in the research literature [22, 27, 49].

One consideration when measuring in-game player behaviour is whether the gameplay is uniform or divided into several distinct modes of play. Much work on adaptive content generation has looked at games where the player is engaged in roughly the same continuous activity, e.g. simple platform games. For such uniform gameplay, player features can be given a consistent interpretation. In other games, gameplay is structured as a series of distinct and possibly overlapping activities. Different player features, such as the rate of weapon fire, can have very different interpretations between activities. For example, *Pac-Man* (Namco 1980) involves a ghost avoidance and a ghost hunting phase, and a "distance to ghosts" feature has a different meaning in each mode. Comparing such features across players may not give us a clear picture of individual differences unless the activity context is taken into account. One approach is to segment gameplay logs into distinct activity types and measure these separately [25].

1.2.2.3 Player Models Player input data can be passed directly to a content generation system, or instead be first converted to a more abstract *player model*, i.e. a representation of the player designed to be more appropriate for subsequent content generation. In general, any representation, e.g. first-order logic, that raw player data is converted to in a preprocessing step could be considered a player model. Typically, this will be a feature-based model, which describes players in terms of small number of features representing significant characteristics.

A feature-based player model consists of scalar *traits* and categorical *types*, following the terminology of personality psychology. A type-only model is known as a *player typology* [4, 5]. Trait-based models are regarded as a more accurate representation of individual differences than discrete typologies, although in some cases a typology may be more convenient for a game designer or content generation system to work with.

Providing that they capture the relevant aspects of the player data with respect to the game, player models can provide a simpler and more convenient representation of the player, considerably reducing the dimensionality of the input data for subsequent adaptive content generation. Ideally, translating player input to a model will highlight relevant variations between players and filter out irrelevant data. If machine learning is used to train the content generation system, working with low dimensional data can increase learning performance. Another advantage of player models is that they

provide a simple representation of the player that can, in some cases, be transferred between gaming contexts, presented to designers and players, or reasoned about by AI agents.

The employ a player model, one must first be created, or selected from a set of existing models. Secondly, a mapping from the player input data to the model must be defined. Finally, the use of the model will need to be evaluated in the current gaming context. Using the wrong player model may lead to useful information about the player being discarded, which will harm performance. Hence questions about the accurate representation of players and the demonstrable benefits for adaptive content generation have to be raised, and the model should be compared to a direct use of player data.

We distinguish between the following three broad types of player model: personality models, experience models and behavioural models, as detailed below. These describe different approaches taken in the player modelling literature, but are not intended to be mutually exclusive or exhaustive.

a) Personality: *Player personality models* describe the player in terms of some general theory of individual psychological differences. Models can be drawn directly from mainstream personality theory, such as the Five Factor OCEAN trait model (e.g. [60, 64]) or emotional valence and arousal traits. Alternatively, they can be applied theories of personality tailored to the gaming domain, such as Bateman's Demographic Game Design typology [5] (e.g. [20]) or Lazzaro's model of emotional motivation [39]. Modern personality models are likely to be supported by evidence for their validity. Data on test players personalities is required to establish the mapping between player data and personality, which could perhaps be carried out using a machine learning approach. In some cases, it may be possible to directly assess players beforehand, e.g. as part of the game [45]. Personality models have the advantage of being transferable between gaming contexts, so information about players can be reused. Conversely, they are somewhat abstracted from players' interaction with a specific game. Adaptive systems can respond to personality models by providing content tailored to the player's estimated personality.

b) Experience: *Player experience models* describe what the player is experiencing during a specified period of play, as estimated from the player input data [63]. Models here tend to be more ad hoc and game-specific (e.g. combat intensity [7]), due to the lack of a generally accepted theory on player experience. However, player experience traits such as engagement and challenge are often used e.g. [46]. Qualitative research into a particular game or genre may provide insight into the experiences to include in a model [26]. As with personality models, data on player experience allows a relationship between player input and experience to be learned. Experience models have the advantage of being highly relevant to adaptation of game content — where the ultimate aim is to improve the player experience – and such models are key to experience-driven procedural content generation [63]. On the other hand, it can be difficult to accurately predict player experience from the available player data, and more indirect methods may better support adaptive content generation.

c) Behaviour: *Player behaviour models* describe what the player has actually done, both within the game and in other domains, e.g. social or physical behaviour. Un-

like personality or experience models, behavioural descriptions tend to be closer to the player input data and, as such, there is potential for a much greater variety of models. Conversely, they are further removed from the player experience that — in an adaptive gaming scenario — the generated content is supposed to enhance. A key advantage is that behavioural models for a specific game can be generated from gameplay data using unsupervised learning. One approach is to synthesize a a low dimensional behavioural model by applying a dimension reduction technique to a sample of high dimensional player input e.g. PCA [49], multidimensional scaling [49, 56], ESOM [23] or player-per-class LDA [25].

1.2.2.4 Measuring Adaptation During the design or evaluation of adaptive gaming systems, measurement of content and players can also be used to evaluate the effectiveness of adaptation. The adaptation requirements describe how player experience should be influenced by changes in game content, and these are generally expressed in terms such as player satisfaction, fun and immersion — although negative experiences may also form part of the requirements, for instance, as part of an engaging and dramatic gaming experience, such as frustration, despair and fear.

We can distinguish between subjective and objective experience measures [62]. Subjective measures ask the player to report their internal experience, and are often categorised as being i) either *quantitative* or *qualitative*, and ii) either *concurrent* or *retrospective*. Quantitative subjective methods (e.g. questionnaires) provide precise, narrowly defined data that is open to statistical analysis. Qualitative subjective methods (e.g. interviews, think alouds) generate richer data which is typically harder to interpret. Concurrent methods collect player reports during play, whereas retrospective methods are used after play.

Methods for subjective measurement can be informed by psychological theories about engaging player experiences, such as:

Challenge, curiosity and fantasy, which are the three main categories in a classic model of fun in instructional computer games by T. Mallone [40].

GameFlow, which comprises eight metrics: challenge, concentration, control, clear goals, skills, feedback, immersion, and social interaction to form a model of player fun [55].

The Player Experience of Need Satisfaction model, which attributes motivational energy in a player to the satisfaction of three basic psychological needs: game competence, autonomy, and relatedness [48].

Objective experience measurement attempts to test the adaptation requirements through unconscious player responses known to correlate with experience. Physiological measurement involves recording body metrics such as heart rate, skin conductivity, breathing rate, posture, jaw muscle tension, or even brain activity. In particular, Mandryk et al. [41] have found that heart rate and jaw electromyography (EMG) correlate to arousal and positive valence in interactive play environments. This approach can be time-consuming and expensive to conduct, and measurements

can be difficult to interpret in terms of conscious experience. However, they eliminate the danger of bias in player reports, such as biases in memorisation and recall of experiences.

Adaptive generation runs the risk of producing entirely unplayable games. This can be limited by a careful selection of the constraints on adaptation and by employing human or automatic playtesting. Alternatively, player expectations can be managed so that occasional low quality experiences are tolerated. Another danger is that players perceive adaptation and react negatively, especially if it is performed on-the-fly or frequently. Changing game content has the potential to introduce confusing or annoying inconsistencies. Such effects can be mitigated by explicitly indicating to the player what has changed and why.

1.2.3 Adaptation

The third step in the game adaption process is to deliver generated content that is personalised to the player. Adaptation can use generated content, measurements of content quality and assessment of the target player to select appropriate content. In some systems, the process ends when an artefact is delivered, such as a complete game. In others, adaptation is an ongoing process, with the player's reaction to the new content continually measured, and further changes made accordingly. Below we consider some general principles for adaptation based on our work.

Adaptation aims to improve player experience. The kinds of experience we want players to have, and the kinds of content an adaptive game provide to provoke those experiences, can vary as much as they do in non-adaptive games. Indeed, adaptive game design faces the same challenges as game design in general. But by handing over responsibility for certain design decisions to adaptive system, we can delay them until the system can take advantage of new data about a particular target player or group of players. The system can make decisions both before play, using preexisting player data, or during play, where data about the current game is also available. Designing an adaptation mechanism can be cast as a machine learning problem, where we need to learn a mechanism that maps player data to game content that satisfies the requirements. Typically, the input and output spaces are categorical and/or scalar features, but in general these can be arbitrarily complex data structures. Player models can be used to simplify player input into a less complex feature set (see section 1.2.2.3). The most common form of adaption is *dynamic difficulty adjustment* (DDA) e.g. [33, 35, 65], where features of the current player performance (player input) are deterministically mapped (mechanism) to a set of game parameters (content output) that affect the level of challenge — e.g. enemy numbers and health in a first-person shooter — so that the game is neither too hard nor too easy (requirements). The option to set a difficulty level can been seen as a simple form of DDA, where the only input is the player's selection from a list, and output is a predetermined set of parameters.

The design of the adaptive mechanism is constrained by the nature of the player input and content output, and the desired relationship between them. If these are relatively simple, as with basic forms of DDA, it may be possible to hand-code a map-

ping. However, more complex forms of adaptation require automated approaches. It is possible to learn direct mappings, using techniques such as multidimensional regression or structured learning. Adaptive rule sets can be generated using reinforcement learning [54, 44]. In some applications, it is easier to recognise suitable generated content than it is to construct one directly — for instance, when designing an adaptation that needs to engage or scare the player. This suggests a generate-and-test approach, where candidate content is created using a *generative* method, such as a design grammar or evolutionary computation e.g. [53], and then assessed against the adaptation requirements using an *evaluation* method, such as a fitness function over a set of player and and content features (see section 1.2.2.3). One such approach that has been successfully applied in several domains is *experience-driven procedural content generation* (EDPCG) [63], where the evaluation step is informed by a player experience model (PEM; see section 1.2.2.3). The PEM predicts player experience from player and content features, and the generated content can be evaluated in terms of how well the predicted experience satisfies the adaptation requirements. See [63] for a definition of EDPCG, and an extensive survey of work in this area. An advantage of EDPCG is that the PEM can be learned from playtest data relating player and content features to player experience [46].

1.3 Applications

In this section, we describe some of the projects related to digital games conducted by members of the CCG, to demonstrate the practical application of the principles described in the previous sections. These include both previously completed projects and ongoing projects still in development, to give an indication of future directions that we may follow; our research itself is constantly adapting to new discoveries, to suit the changing needs of the field. While we are interested in raising and answering generic questions related to adaptive game design, our individual projects have tended to work with games of one genre. We cover six types of games, and it seemed sensible to break down our work below into the types of games with which we have experimented. In subsection 1.3.1, we look at 3D action games, with an emphasis on player modelling, followed in subsection 1.3.2 by an investigation of arcade games, with an emphasis on all aspects of the adaptive game generation process: generation, measurement and adaptation. In subsection 1.3.3, we concentrate on generative processes in games of the platformer genre, and we look at generative and measurement aspects of the adaptation cycle in subsections 1.3.4 and 1.3.5 in the context of board games and puzzles respectively. Finally, in the context of open-world games, we look at adaptation of games before they are played in subsection 1.3.6. Table 1.1 summarises these applications and the approaches used for each.

1.3.1 3D Action Games

First- and third-person action games allow the player to explore complex 3D environments and engage in activities such as exploration, combat, acrobatics and problem

Application	Generation	Measurement				Adaptation
		Content	Player			
			Personality	Behaviour	Experience	
3D Action						
The Hunter	—	—	—	Yes	—	—
Rogue Trooper	—	—	—	Yes	—	—
Arcade						
Pac-Man	—	Yes	—	Yes	—	—
Super Mario	Yes	—	—	Yes	—	Yes
Platform						
ANGELINA	Yes	Yes	—	—	—	—
Board						
LUDI	Yes	Yes	—	—	Yes	—
Shibumi	Yes	Yes	Yes*	—	Yes*	Yes*
Puzzle						
Hour Maze	Yes	Yes	—	—	Yes	—
Open World						
Subversion	Yes	Yes	—	—	Yes	Yes

Table 1.1 Approaches used per application (* denotes upcoming work).

solving. Players are typically free to move around the environment and engage in different activities as they choose, limited by factors such as their location and the behaviour of other in-game entities. This high level of non-uniformity of behaviour and experience in both space and time presents challenges for player measurement: how should we compare players who have chosen different paths and activity schedules? This is particularly challenging in open-world games, where player behaviour is relatively unconstrained. We look below at two attempts to generate player behaviour models from gameplay logs (see section 1.2.2.3) which account for such large variations in gameplay.

1.3.1.1 Modelling player exploration In 3D environments where the player is free to move around, analysing how they have explored their environment could be a useful component of a player model, and used to inform subsequent content generation. Ramirez-Cano et al. [49] generated a behavioural player model for *The Hunter* (Expansive Worlds 2009), a realistic first-person hunting game, based on gameplay logs from approximately 50,000 players. Their model combines measures of in-game actions and performance, level exploration and use of a game-related proprietary social network — here we look at their analysis of player movement.

The Hunter allows players to move around a large rural environment, tracking simulated wild animals, then shooting or photographing them. The game periodi-

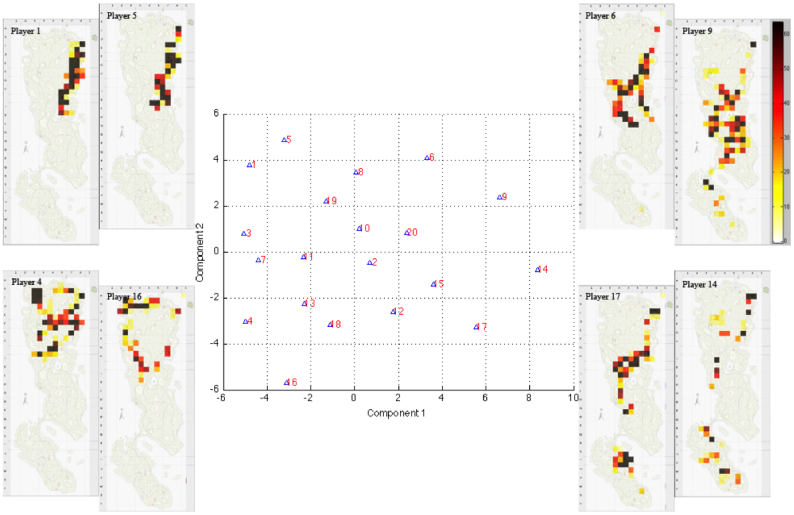

Figure 1.3 A 2D representation of 25 players' heatmaps, generated from Earth Mover's distances between heatmaps using multidimensional scaling. Examples are shown for some nearby pairs of players. Reproduced from [49].

cally logged each player's location, recording their path through the level as a sequence of coordinates. In the level studied, players explored a large island. The level map was divided by 2D grid, and a 2D location heatmap could be generated for each player by summing the time they spent in each grid square. Figure 1.3 shows some examples of player heatmaps. Heatmaps were compared between players using the Earth Mover's distance (EMD): this assumes a fixed cost for moving one unit of distribution mass from one grid square to an adjacent non-diagonal grid square; the distance between two heatmaps is then the minimal cost for a set of moves that transforms one map to the other. Optimal EMDs were calculated using Rubner's algorithm. The heatmap EMD provides a measure of dissimilarity between two players' exploration paths.

To generate a low-dimensional model of how player exploration varied, a dissimilarity matrix was computed for a random sample of 20 players, giving the EMD between each pair of players. Multidimensional scaling was used to reduce this matrix to a 2D representation [8], shown in Figure 1.3. The figure illustrates that players who are located near each other in the 2D space have similar exploration paths. This gives a two trait model of player movement within this level, which allows an easy comparison of players' exploration activity. Although interpreting the traits is difficult, players who score highly on the vertical trait spend a lot of time on the right of the map, whereas those who score high on the horizontal spend time at the bottom of the map. Irrespective of interpretation, it provides a convenient representation of exploration on which to base further content generation. For this type of player model

to be used for adaptation, new players need to be assigned trait scores. Methods exist for defining Multidimensional Scaling (MDS) axes in terms of known features, but a simpler approach is to treat the player sample as a set of prototypes: for a new heatmap we find the nearest prototype heatmap by EMD, and use its trait scores. Alternatively, a typology of exploration types can be generated by clustering players in the MDS space, and new players are assigned the type of the prototype with the nearest heatmap.

In general, using a player sample to construct this kind of model is justified by the high complexity of MDS, although modern variants have lower complexity and should be able to cope with a larger numbers of players. The choice of the number of MDS dimensions (traits) is also critical, and scree and stress plots can be used to compare low-dimensional representations [8].

This use of EMD and MDS allowed us to find a simple representation of complex player exploration paths, and hence compare players' exploration behaviour. This could be valuable information for an adaptive game, as it could reflect players' style and exposure to level content. In [49], we combined this data with measurements of the in-game actions of players and related social-media activity, to create a rounded player model for *The Hunter*.

1.3.1.2 Modelling player combat Player behaviour in complex games can involve multiple activities, and player modelling can be improved by segmenting gameplay into separate activity phases, and analysing these individually. Gow et al. [25] present a trait-based behaviour model of player combat activity in *Rogue Trooper* (Eidos 2006), a third- person shooter, generated from gameplay logs from 32 players. Each log recorded a playthrough of the game's first level, until they completed it or quit, with a mean play time of 18 minutes — in total, over 10 hours of logged gameplay. Players were observed engaging in a variety of activities: preparing for and engaging in combat, checking areas for remaining enemies, fleeing and avoiding enemies, navigating to and exploring locations, getting lost, retracing their steps, investigating the controls and game mechanics, even admiring the scenery.

To compare players, we decided to focus on combat, the central activity in the game. A player combat behaviour model was automatically synthesised from the gameplay logs, consisting of three traits: *dynamism*, *cautiousness* and *ammunition management*. As a further abstraction from the data, a player typology was generated from these two traits, consisting of four combat behaviour types: *hyperactive* (high dynamism), *normal* (medium dynamism), *naive* (low dynamism, low cautiousness) and *timid* (low dynamism, high cautiousness). Figure 1.4 shows the 32 players' dynamism and cautiousness scores, and player types. The trait-based model was computed in a four stage process as follows:

Segment activities Identify sections of combat activity in log data, and extract individual combat instances. Each instance was defined as starting whenever a distinct group of enemy Non-player Characters (NPCs) fired on the player or were shot by her, and ended when either the player or the NPCs were dead. Only successful combat instances were used to construct the model.

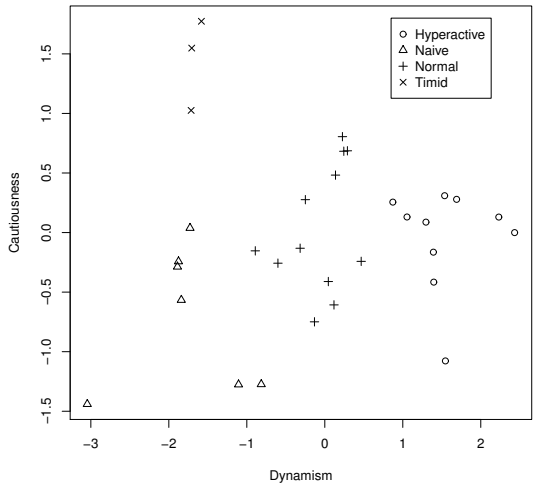

Figure 1.4 A two trait player combat behaviour model for Rogue Trooper. Individual players are shown with their combat behaviour type. Adapted from [25].

Define features Calculate 21 scalar features for each combat instance, e.g. proportion of time using each weapon in addition to taking cover, rate of fire, movement, rotation, mean ammunition and health.

Compute player discriminants Partition combat instances into classes, one per player, then use LDA to compute a series of linear discriminants (linear combinations of features), i.e., directions within the 21-dimension feature space which maximise between-player variance and minimises within-player variance [43].

Select and interpret traits Choose a small n, such that the first n discriminants account for much of the variance between players and can be interpreted as meaningful traits. For this analysis of Rogue Trooper combat, the first three discriminants were selected.

Calculate trait scores Each player's trait scores are defined as the centroid (mean position) of their combat instances in the trait space.

Finally, the player typology was generated by applying K-means clustering to the players based on the dimensions of the first two discriminants.

Segmenting gameplay into distinct combat and non-combat phases allowed us to identify meaningful player combat traits, which reflect their style of play. Similar features of their behaviour over the entire level would have obscured their playing style, due to huge differences in activity phases between players. The work also demonstrated the utility of using one-player-per-class LDA as an unsupervised

learning technique to find interesting behavioural differences between players. In related work, we have looked at how variations in gameplay during combat activity influences player experience [26]. Other activities, such as problem solving and exploration, could also be extracted and compared using a similar approach. One future direction for research is to investigate improved techniques for reliably segmenting game logs into activity phases and identifying activity types. As a final point, we note that it would be straightforward to use either the trait model or typology in an adaptive context: once a new player has completed a combat task, their trait scores can be calculated as a linear combination of a few simple combat features.

1.3.2 Arcade Games

Arcade style action games are descendants of early examples of video games originally installed in coin-operated entertainment machines in amusement arcades. The genre, which had its so called Golden Age in the early 1980s with games such as *Pac-Man*(Namco, 1980), *Space Invaders*(Taito, 1978) and *Donkey Kong*(Nintendo, 1981), enjoys a recent resurgence with newly released games styled to emulate the visuals and game-play of the arcade classics. This renewed interest mainly stems from factors such as the simple and easy to learn game-play suitable for mobile devices and a relative ease of development especially for very small development teams, which have found increasing support and foothold in online distribution markets for consoles, PC and mobile platforms. The relative simplicity both in visual style and game rules encourages quick prototyping of game mechanics and style, and makes it an interesting target for player profiling and game adaptation. We discuss below a trait-based adaptation mechanism that we have experimented with.

1.3.2.1 Trait-Based Adaptation Arcade games, with their limited size of game environments and game rules that are often based on very simple mechanics, feature a relatively homogeneous game-play, especially when compared to story-driven or exploration-based games. This homogeneity allows for a simpler trait-based player behaviour model that does not need to segment game-play into (many) different phases or concern itself with breaking down behaviour to match certain plot sequences as it might have to in a story-driven game. Additionally, a homogeneous model is beneficial for game adaptation, as there is no need for a mapping of game phases or action sequences to measured traits.

Conversely, the complexity of game adaptation according to traits can be reduced in a game with such sequences if each game sequence is handled as a separate optimisation instance. However, traits that are affected by multiple such sequences, or are a result of the order of sequences, cannot be adequately handled in such a separation. For example, an action sequence might be followed by a puzzle sequence in shooter games such as the *Half-Life*(Valve, 1998, 2004) and *Tomb Raider*(Eidos Interactive, 1996–2010) games, giving the player time to recover and relax, which slows down the pace of the game. Simply handling these sequences separately would lead to an inability to recognize a player's reaction to this pacing. More advanced

techniques such as observing changes in traits over sequences and cross-validating adaptations in each sequence can be used to alleviate these difficulties.

To create a trait-based adaptive game, we propose the following structure and outline experiments we have conducted following these steps. Note that the first three steps have also been outlined in Section 1.3.1.2, in a different context.

Define features Select player metrics such as input rates, actions and events. Optimally, these metrics would be time-independent rates to allow for recordings of different lengths, e.g., by recording actions-per-time, such as *Pill per minute* in *Pac-Man*(Namco, 1980). Due to the nature of the discriminant analysis (see below), which automatically identifies significant features, a large amount of metrics can be recorded.

Compute player discriminants Use Linear Discriminant Analysis (LDA) to compute a series of discriminants (linear combinations of features) — directions within the feature space which maximise between-player variance and minimises within-player variance.

Interpret traits LDA generated feature vectors are sorted by importance, i.e., how much of the variance between players can be explained by them. This property is effectively compressing the data and allows for a dimensionality reduction of the LDA-transformed feature space with low information loss. Furthermore, it allows the researcher to quickly assess the most influential metrics on player separation, as they will appear in the first LDA vectors.

Correlate traits with preferences Once important traits have been identified, they need to be correlated to player preferences. This can be done explicitly through surveys that evaluate player preferences on a series of selected rule-sets, or by manually connecting traits with game features by the game designer, or by using some model that associates traits with preferences through psychological models. In our experiments, we have focused on the former two approaches.

Trait-based adaptation Finally, the game is adapted for a player by first measuring their traits, followed by generating a matching rule-set through retrieving preferences of similar players, compared by traits. This can be done in a variety of ways, for example by using a K-nearest neighbour search in the LDA-transformed feature space, effectively measuring a trait distance.

1.3.2.2 Results from LDA Analyses Several experiments have been conducted to record and interpret player traits in arcade games. Using a re-implementation of *Pac-Man*(Namco, 1980), Baumgarten [6] applied multi-class LDA to a dataset of 245 players, each of which played 5 or more rounds of the game. Features recorded in the game include player actions: eating a pill, eating a power-pill, eating a ghost, getting eaten by a ghost, eating a fruit, finishing a level, and changing the movement direction; rate of player actions: pills per minute, ghosts eaten per power pill, ghosts eaten per minute, power pills per minute, key strokes per pill, changes in movement

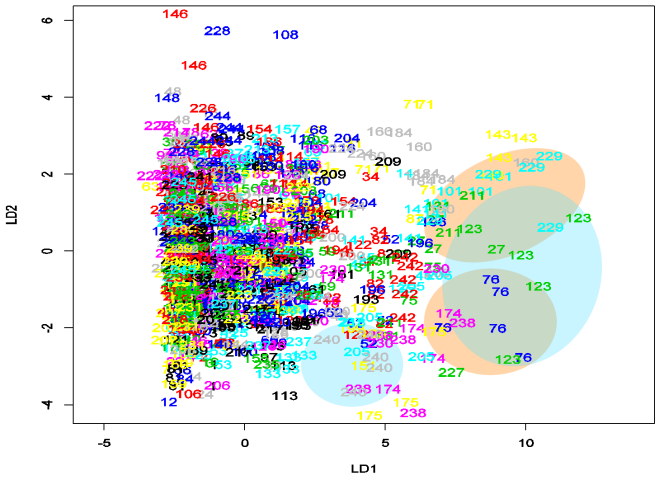

Figure 1.5 LDA transformed space of metrics recorded in *Pac-Man*. Each number represents a player, and each number will appear five times, its position marking the coordinates of the feature vector defined by one session of that player. Some player sessions have been highlighted by colored circles. Reproduced from [6].

directions per pill. A projection of the first two dimensions of the resulting player discriminant space is shown in figure 1.5.

The most insightful tool to interpret the traits generated by the LDA algorithm is analysing the weights of the base vectors defining the LDA-transformed space. These vectors define the mapping of player metrics into the LDA space, where each weight associated with a player feature indicates its influence in separating players in that dimension. In the *Pac-Man* survey, the single largest weight for the first LDA vector was given by the *keystrokes per pill* metric, which indicates the efficiency of a player to navigate the game with respect to the physical interaction with the computer. The second dimension was dominated by negative weights on the number and speed of power pills eaten and positive weights on ghosts eaten and number of turns. This relationship models a game concept of *Pac-Man*: chasing ghosts with a high efficiency by eating many ghosts while not using up many power pills. In short, the most distinguishing feature of the game was physical interaction, followed by skill at playing the game well.

We have followed up this research with a new arcade-like game called *Snakeotron* that is inspired by a light-cycle race in the movie Tron (Disney, 1982), as described in [25]. We have found comparable results with the linear discriminant analysis highlighting metrics related to physical interaction being found to be the most distinguishing traits in the game, followed by traits describing skill — in *Snakeotron* this relates mainly to avoiding walls in the level. *Snakeotron* has been designed to adapt

to individual players, as they progress, in response to feedback. The adaptations include aspects of difficulty and aspects of gameplay, e.g., the ability to jump over lines, rather than just avoiding them. We have carried out a first set of experiments with *Snakeotron* which has included a questionnaire after each round with a different rule-set. Players were asked to rate their enjoyment relative to the previous level. Such pairwise comparison is more accurate and robust than absolute measurement [61]. In future work, we plan to use a machine learning algorithm such as reinforcement learning or a neural network to predict player preference values for new rule-set combinations. This approach can be further augmented by taking into account other players with similar traits to increase prediction accuracy, and to eventually generate preference values with only trait data available. Ultimately, we plan to show that dynamic adaptive personalisation is possible, albeit with a simple arcade game.

A machine learning approach may not strictly be required here, as hand-crafted adaptation according to measured traits is often more practical in commercial game development. One such hand-crafted adaptation approach has been implemented in an adaptive *Super Mario*(originally Nintendo, 1985) level generator. As part of a Level Generation competition by Shaker et al. [52], which asked participants to submit programs that take metrics of a single test-run of players and generate a level for a variant of *Super Mario*, we developed a small level generator implementing trait-based adaptation as described above. LDA has been used to analyse the data from play-throughs collected during a pilot study. The first LDA discriminant vector emphasised metrics relating to the time required to complete a level and the number of jumps, which can be interpreted as the ability to complete levels quickly and efficiently. This dimension was thus used to judge the player skill and to adjust the subsequent levels by generating a level made out of a selection of hand-crafted building blocks. These building-blocks were previously annotated with a difficulty rating and picked according to the relative skill of the player — the more skilled the player is, the more difficult building blocks were chosen.

1.3.3 Platformers

Two-dimensional platformers have surged in popularity over the last five years thanks to many open game libraries tailored towards making them[1] as well as the relative breadth of subgenres, from puzzle-based platformers like *Offspring Fling* (KPULV, 2012) to large-scale Metroidvania games like *Saira* (Nifflas Games, 2009) or *Knytt Stories* (Nifflas Games, 2007). The constrained two-dimensional space simplifies almost all aspects of implementation and design, making it highly amenable to Artificial Intelligence techniques. We describe here an evolutionary approach to generating fully formed platformer games.

ANGELINA is a co-operative co-evolutionary system for automatically designing simple Metroidvania-style platform games. This subgenre of 2D platformer puts an emphasis on exploration, where new areas of the game world are made available as

[1]For instance, see Flixel (http://www.flixel.org) or FlashPunk (http://www.flashpunk.org)

Figure 1.6 An outer template blocking left, right and down exits.

Figure 1.7 A mask blocking the top exit with special blocks.

the player gains abilities and collects various items. As described in section 1.2.1.3, co-operative co-evolutionary systems are composed of several evolutionary subsystems that are brought together in order to evaluate the quality of their populations. This section briefly describes the composition of ANGELINA's subsystems and how the fitness functions evaluate each component, in addition to how ANGELINA integrates sound and images mined from the internet in response to topical social media. For a more detailed description of the evolutionary system, see [18], and for more details of the social media subsystem, see [19].

1.3.3.1 Evolutionary Design Subsystems The evolutionary subsystem for level design maintains a population of $Level$s, which contain two-dimensional integer arrays that represent the tiles that make up the game world, and a *collision index*, c, such that any tiles with an integer value $\geq c$ are initially solid and not passable by the player. Rather than placing every tile individually, levels are built out of screen-size groups of tiles, by default 20 tiles wide by 15 tiles high, called *chunks*. Chunks are generated by combining an *outer template* (defining which edges of the chunk are passable), an *inner template* (defining what tiles are in the core of the chunk) and one or more *masks* which add extra blocking tiles to the chunk to allow for features like locked doors. Figure **??** shows an example of each of the three components that make up a map chunk. ANGELINA uses all permutations of outer templates and masks, and has 16 hand-designed inner templates. Chunks are randomly generated by combining these together, and arranged in a grid layout to create a map. The fitness function for the level design subsystem prioritises high reachability maps (using a simulation of the game physics in order to estimate which areas are accessible in a given map), as well as scoring highly those levels which increase the travel distance between powerups, the exit, and the player starting point (which change during evolution, and are supplied as part of the co-operative co-evolutionary process during evaluation).

The layout design subsystem creates mappings that define the placement of the player starting point, the game exit and the enemy types into the game world. ANGELINA's evaluation techniques don't calculate the difficulty of combat, which means that enemy placement and design is relatively simple and underused - enemies are

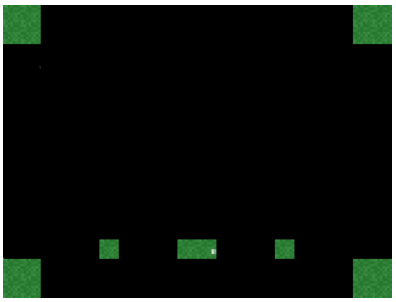

Figure 1.8 An inner template with blocks providing the chunk's content.

Figure 1.9 The completed block, composed of masks, inner and outer templates.

designed by combining predefined behaviours and then associating that design with a set of co-ordinates that state where enemies of that type exist in the world. The fitness function is weighted to discourage overcrowding of enemy placements, but better combat simulations could take into account the change in difficulty as the player progresses through the level. More important is the placement of the player start and the game exit, as they define the flow of the game. As with level design, reachability maps are supplied to the fitness function when evaluating layouts, which give information on shortest paths and the percentage of the game level that is explored. The fitness function assigns higher scores to longer paths.

A powerset is a set of powerup items, along with their defined placements in the game world. Powerups are items that change the abilities of a player, enabling them to access new areas of the map. A powerup is defined by a game variable it is attached to, and a value it sets that variable to when the powerup is collected. In the powerset evolutionary subsystem, for simplicity, ANGELINA can assign a powerup to one of three variables - the in-game collision index (defining which tiles are solid), the world gravity (affecting the player and all enemies and their reachability), and the player's jump power. Values are assigned randomly within wide, but reasonable, boundaries. The specific values a powerup has are fine-tuned as part of the evolutionary process, meaning ANGELINA can give the player precise bonuses to their jump height that allow them to reach some new parts of the map, while leaving others out of reach. This often happens as part of the evolutionary search, as it increases the paths through the level, which results in higher fitnesses for layouts and maps.

The fitness function for powersets considers the placement and proximity of powerups to each other (again rewarding powersets which increase the travel for the player) as well as looking at possible traces through a game. In some cases, powerups can only be collected in a single linear order. For instance, Figure 1.10 shows a linear progression where the player starts at (1), then collects a jump powerup (2), which enables the collection of a key (3) to unlock the path to the game exit. Note how the powerups encourage movement across the map; (3) is on the other side of the game world to the exit, (4), for instance.

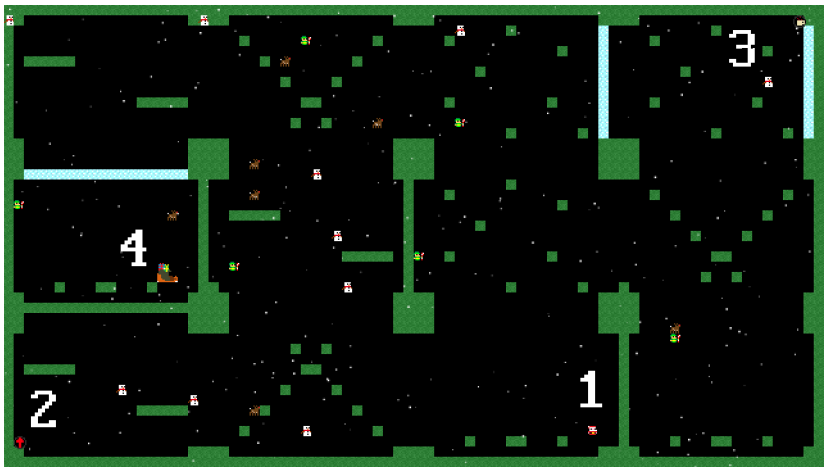

Figure 1.10 A full game map showing player progression.

In other cases, powerups may be partially ordered instead and multiple paths may exist to the exit. ANGELINA penalises powersets that are very nonlinear, and in doing so encourages a stricter path through the game which makes other elements of design (such as the optimisation of shortest routes) easier. It also rewards powersets that provide the player with a steady sense of progress, rather than powersets where initial gains are very large, and then the rest of the game is spent collecting very small increases in ability.

1.3.3.2 Social Media When we talk about generating adaptive or bespoke game content, we often think of content that is generated as a result of examining a specific person's preferences and behaviour. An alternative is to adapt to specific cultures, countries, periods of modern history, or groups of people. Adapting content on a broader scale is another way to make games more reactive, more interesting and more meaningful for players. To explore such possibilities, we augmented ANGELINA with the ability to make simple platform games about topical news stories. In particular, in this modality, ANGELINA starts from a single piece of source data, namely a newspaper article from the Guardian (UK newspaper) website, and expanded the data to produce enough information to form the theme of a game.

The Internet currently contains a wealth of open data of all types, including music, sound clips, video, photographs, poetry and many other resources. However, automatic downloading and usage of these resources is difficult, due to a lack meta-information about what is currently being looking at and what data is actually required. The first problem – that of understanding the data you are working from – should be solved with as little reliance on unknown information as possible. ANGELINA's starting point, the Guardian article, is often full of complex linguistic information, turns of phrase, and oblique references to other news articles that may

not be linked or understood by ANGELINA. While techniques do exist to extract information from bodies of text like this, we opted for a simpler approach that takes advantage of the richness of data the web contains. ANGELINA takes words from the headline, and the tags the Guardian writers associate with the article, and uses this to build a list of keywords for the news article. These keywords are then used to search for more content from the web.

While we could use the keywords as they are, we have found it useful to first classify them as referring to a person, referring to a country, or neither. Tools like WordNet[2] and ConceptNet[3] are open platforms that can help connect one word with other words and phrases that are related, as well as giving information on what it is that relates them. Wikipedia is also useful for collecting large amounts of viable data and keeping it regularly maintained. While large portions of the site may be unreliable and open to vandalism, many pages are well-curated and closely updated, providing useful contextual information that may not be available anywhere else. ANGELINA uses the *List of sovereign states* to confirm whether a keyword describes a country or not, and checks to see if Wikipedia has a page about a keyword to determine if it refers to a person. If a page exists, ANGELINA simply checks for information that confirms that the page refers to a person – data fields like birth year, or nationality. These checks are not completely robust – victims in a murder inquiry, for instance, may not have Wikipedia pages – but it provides a useful shorthand that guides ANGELINA in the next step – finding usable and relevant content online.

Given the partially classified keywords, ANGELINA searches Flickr and Google Images for general photographs using them as search terms, and uses these images throughout the finished game. It also uses the Incompetech[4] website, which hosts a large collection of Creative Commons music, tagged with moods and genres. The tags allowed ANGELINA to select music based on the feeling it might convey, and is reliably curated by the composer. Using the AFINN database[5] of emotionally-tagged words allowed ANGELINA to guess at the tone of the article, and chose sad or happy music accordingly. ANGELINA also uses the sound effect database FreeSound[6] to search for sound effects using article tags. This can have mixed results, because the submissions are described and categorised by the individual submitter, and only rough metrics (like number of downloads) exist to tell recordings apart. This can produce powerful results – like gunfire and historical speeches being returned for searches about war – but can just as easily return offbeat or inappropriate results too. Often, the rewards outweigh the risk, particularly when the potential for surprise or unexpectedness can be woven into part of the adaptive system's appeal.

Figures 1.11(a) and 1.11(b) show screenshots from games created using this system, utilising photographs and images from Google and Flickr, backed with a soundtrack and an array of sound effects. Note that the title of each game is also created

[2]http://wordnet.princeton.edu/
[3]http://conceptnet5.media.mit.edu
[4]http://www.incompetech.org
[5]http://fnielsen.posterous.com/tag/afinn
[6]http://www.freesound.org

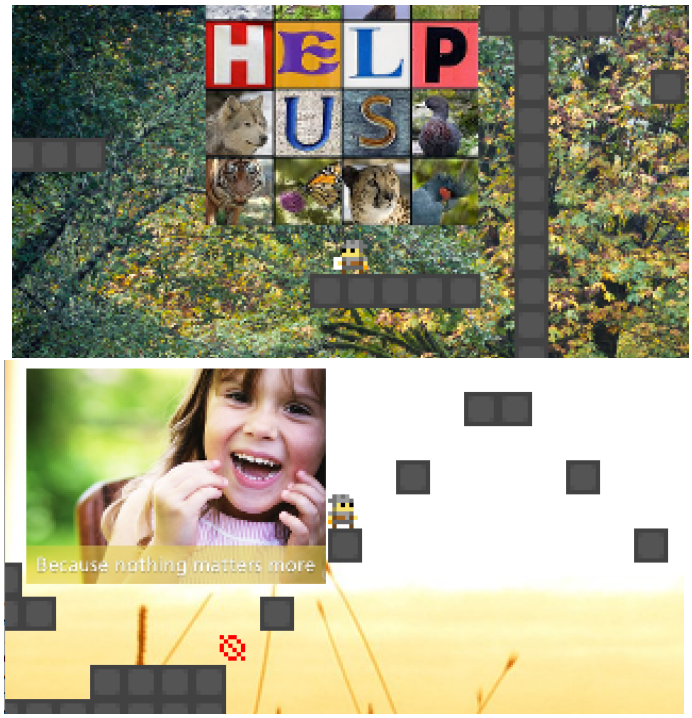

Figure 1.11 (a) A screenshot from *The Conservation of Emily*, a game based on a news article about illegal logging in South America. The background image is obtained from Flickr searches, and inset image from Google Image searches. (b) A screenshot from *Sex, Lies and Rape*, a game based on a news article about a child abuse ring. The game features a sad musical track, and includes a sound recording of a woman singing a children's song in Greenlandic, retrieved from a FreeSound search. Both games are available to play online at www.gamesbyangelina.org/games.

automatically using a keyword search through online rhyming dictionaries for potential puns. The resulting games are currently a little rough around the edges, but they do demonstrates the strength of the approach described above. We are currently adding more sophistication to the game generation process, in particular to use gameplay elements to further illustrate the news story, for instance, interacting (shooting, etc.) characters from the story, solving puzzles to illicit further information about the newspaper article, etc. Following this, we plan extensive experimentation to determine player reactions to such games, ultimately aiming to improve the generative process to a stage where people take their daily news through ludic interaction rather than passive reading. We also plan to enable users to adapt the game generation process, for instance by providing links to the newspaper articles themselves, or supplying their own keywords. We further plan to investigate the potential for personalised games to be generated from social media assets, such as FaceBook pages.

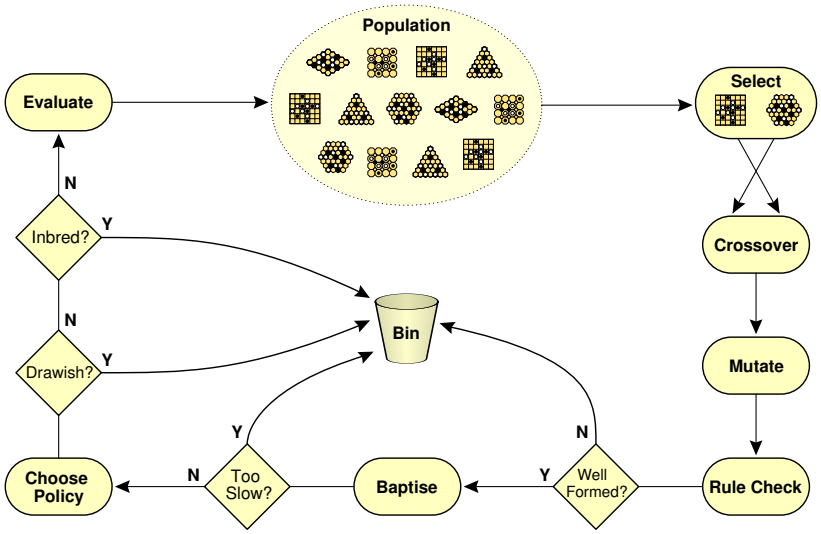

Figure 1.12 Game life cycle (from [10]).

1.3.4 Board Games

A defining distinction between video games and board games is that rule sets in video games are fluid and may be updated on-the-fly, whereas rule sets for board games are typically fixed once the game is published and released. There are of course exceptions, such as the games *Eleusis*[7] and *Zendo*[8] in which one player invents a set of rules while the other players must deduce them through play, and there is also the tendency for players to extend the shelf-life of board games by inventing variants to play [51]. However, in each of these cases, the scope of variation is governed by the resources at hand, and these games are still played within a defining framework. Adaptation in the context of board games can involve the optimisation of rule sets to achieve a desired behaviour within a given system, or the invention of completely new games and even game systems.

1.3.4.1 LUDI LUDI is a system for playing, evaluating and generating new combinatorial board games [10]. Each game is modelled as a LISP-like *symbolic expression* or *s-expression* that defines its rule tree, and new games are generated using standard *genetic programming* (GP) operators of crossover and mutation [38]. Games are evaluated according to certain aesthetic criteria measured over self-play trials, as outlined in Section 1.2.2.1. The weighting of each criterion — indicating how relevant it is to the game's quality — was determined by correlating aesthetic

[7] http://www.boardgamegeek.com/boardgame/5217/eleusis.
[8] http://www.boardgamegeek.com/boardgame/6830/zendo.

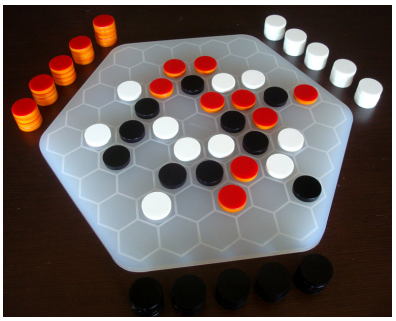

Figure 1.13 *Yavalatah Deluxe* from Nestorgames (for three players).

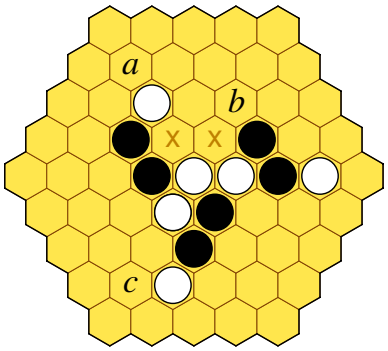

Figure 1.14 A puzzle with white to play.

measurements for a set of source games with human player rankings for those games, giving an *aesthetic policy* for that set of players.

The game creation process used by LUDI is very much an example of search-based procedural content generation, and is summarised in Figure 1.12. It is adaptive in the sense that it biases its search for new games towards those games that score more highly on the aesthetic policy obtained for the target set of players. This value was fixed in the LUDI experiment, but could easily be modified to respond to players' reactions to new games on a game-by-game basis, hence redirecting the search as new games are invented and played. The games invented by LUDI could not be said to be truly *bespoke*, beyond the fact that they were invented specifically to maximise the level of interest in that specific set of test players. LUDI proved successful in this task, producing several new games that interested human players, two of which have gone on to be commercially published.

Yavalath is the most popular game designed by LUDI (which also names its games). It proved popular with the test players, and has been received well in the broader game-playing community following its publication by Nestorgames,[9] for whom it remains a flagship product (Figure 1.13(a)). *Yavalath* appears to have captured some general principles of game design that go beyond the group of test players, as it has since been ranked in the top #100 (or top 2.5%) of abstract board games ever invented, according to the BoardGameGeek (BGG) database, the world's foremost online board game community [12].

The rules devised by LUDI for *Yavalath* are as follows:

```
(game Yavalath
    (players White Black)
    (board (tiling hex) (shape hex) (size 5))
    (end
```

[9] http://www.nestorgames.com.

```
            (All win (in-a-row 4))
            (All lose (in-a-row 3))
        )
    )
```

Two players, White and Black, take turns placing a piece of their colour on the board. A player wins by making 4-in-a-row of their colour, but loses by making 3-in-a-row beforehand. The simplicity of the rules makes the game easy to teach to new players and immediately accessible, but the "win with 4 but lose with 3" rule combination hides an emergent twist that keeps players interested, as sequences of forced moves can be perpetuated using 4-in-a-row threats to manipulate the opponent into a losing position, with clever play.

For example, Figure 1.13(b) shows a *Yavalath* puzzle with White to play and force a win (*hint:* if Black is allowed to play at either point X, then White is forced to play at the other X and therefore lose, hence White *must* play a sequence of forcing moves starting at point *a*, *b* or *c*). The discovery of this forcing move mechanism provides a satisfying "aha!" moment for each individual player, and is exactly the sort of emergence that we had hoped would develop from simple rules in the evolutionary search.

1.3.4.2 Shibumi The *Shibumi* project continues the work pioneered with LUDI, but in this case the design space is constrained to a *closed* game system with a fully defined rule set, in order to compare the search dynamics of human and computer game designers. The *Shibumi set* was designed specifically for this purpose, and consists of a 4×4 square grid of holes and 16 balls in each of three colours (Figure 1.16(a)). 30 balls may be stacked on this board to form a *square pyramidal packing*, as shown in Figure 1.16(b). This system has the computational advantage that each game state can be bit-packed into a single 64-bit `long`.

The term *shibumi* comes from Japanese aesthetics and means *tasteful elegance* or *simplicity hiding complexity*, a very apt notion for abstract board games [11]. We seek the simplest rule sets that produce the most interesting games for this minimal system. The *Shibumi* project is currently a work in progress. A game design contest called the *Shibumi Challenge* has been run, with 45 entries solicited from 22 human game designers. The results, presented in [16], provide an *inspiring set* of source games for automated search, and ensure that the rule set for this system is fully defined. The next step will be to conduct an automated search for new games, using both evolutionary and *Monte Carlo tree search* (MCTS) methods [15], and to compare the dynamics of the search between these approaches and the approaches used by the human designers. We are particularly interested in whether the automated search will find hidden gems missed by human designers, the frequency with which these occur, and the margin by which they were missed.

The ultimate aim is to produce a software tool that is a creative collaborator in the game design process, which will adaptively help the player find the ideal *Shibumi* game for them. For example, the player might specify what rules or features they like or dislike in a game, or simply rank some test games and let the system deduce these preferences. The system will then use this information to search its database of

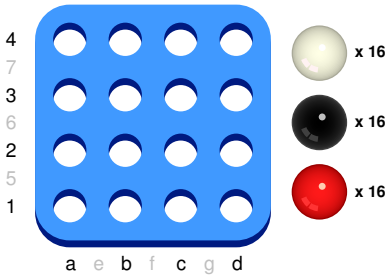

Figure 1.15 The *Shibumi Set*.

Figure 1.16 The Nestorgames edition.

known games, and to create new bespoke games for this player in accordance with these preferences. The player's reactions to these bespoke games can then be used to adaptively modify the search for further games.

We are particularly interested in questions of computational creativity, and the closed and fully defined — yet largely unexplored — *Shibumi* search space provides an ideal test bed for this [14]. It is obvious from observing human designers that the constraints imposed by such a closed system have a significant impact on the creative process (the search is necessarily more *combinatorial* within the design space rather than *transformational* between this space and others), but that creativity can still be achieved through unexpected and serendipitous rule combinations. It is important to maximise the creativity in the process, both from the designer's perspective (so that novel, high quality artefacts are produced) and from the player's perspective (to increase their appreciation of the generated artefacts).

1.3.5 Puzzles

Solitaire (single-player) puzzles can be described as a form of play that is fun and has a right answer [37], or as rule-based systems, like games, in which the goal is to find a solution rather than beat an opponent [21]. A defining feature of puzzles is their *non-replayability*, as once a challenge is solved then it is no longer interesting to the player until they forget that solution [59]. PCG offers a way to quickly and cheaply generate a large amount of new and interesting challenges, to keep puzzles replayable. Solitaire puzzles can also be viewed from a combinatorial game perspective as two-player games that provide a contest between the *designer* who creates the puzzle and the *solver* who attempts to solve it. The designer constitutes a *null* player who may not be physically present for the contest, but whose wit and personality can be evident in the challenges that they set the solver. It is this feeling of intelligence and playfulness that distinguishes human-designed puzzle content from computer-generated content, and is what we would ideally like to capture using PCG.

Japanese logic puzzles are a popular type of solitaire puzzle, following the meteoric rise of *Sudoku* in the western printed media over the last decade. This rise to prominence coincided with the advent of the smart phone as an everyday item,

Figure 1.17 *Hour Maze* for iPad.

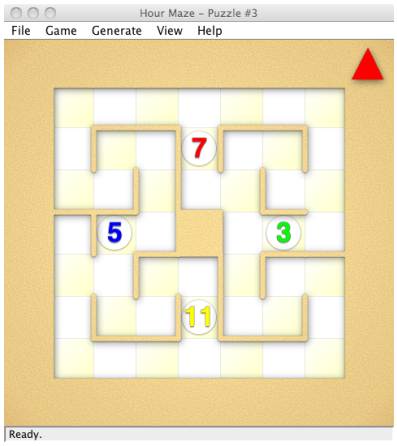

Figure 1.18 A Java-based survey application for Hour Maze.

cementing the popularity of such logic puzzles. They are self-contained, can be effectively played on small screens, and do not require an excessive investment of attention or time; they are in many ways the ideal application for handheld devices and commuters seeking distraction. Japanese logic puzzles are characterised by the following traits: (i) Single player (ii) Simple rules (iii) Unique solution (iv) Can be solved by deduction (v) Context free, i.e. universal symbols such as numbers, not letters or words.

The appearance of *Sudoku* on a mass scale, as a regular feature in UK newspapers starting in 2004, was made possible by Wayne Gould's computer programme POPPACOM SUDOKU, which was designed specifically to generate mass *Sudoku* content for the global market. Gould reported earnings of over a million dollars in less than a year from POPPACOM SUDOKU, and went on to become named as one of the "World's Most Influential People" by *Time Magazine* in 2006.

However, Japanese publisher Nikoli — the primary source of Japanese logic puzzles such as *Sudoku*, *Kakuro*, *Slitherlink*, and so on — remain adamant that human-generated puzzle designs are superior to those generated algorithmically [30, 36], and that a true puzzle afficianado can easily distinguish the two. Nikoli remain distinct from most other publishers in the world by preferring not to release computer-generated content, despite the proven convenience and cost-effectiveness of doing so. We describe below ways in which design features of human-designed puzzles may be incorporated into procedurally generated content, in an effort to reduce the perceived quality gap.

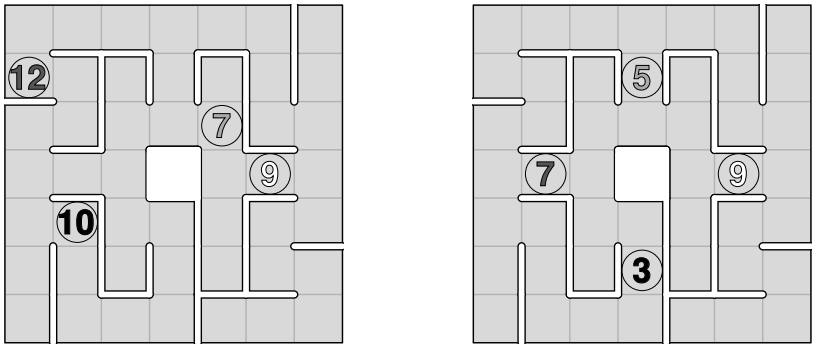

Figure 1.19 The effect of hint symmetry (from [13]).

1.3.5.1 Hour Maze *Hour Maze* is a solitaire logic puzzle game devised by Mike Reilly and released for the iOS platform in 2011 [13]. Figure 1.18(a) shows the iPad version in use. Players solve each challenge by filling the given maze with contiguous runs of coloured *hour sets* $\{1, \ldots, 12\}$, such that each colour set is connected and the difference between each pair of adjacent numbers is exactly 1. One number of each colour is revealed at the start of each challenge as a *hint*. As the puzzle was newly invented in its current form, there did not yet exist a database of challenges, or any human experts on hand to design such levels, so PCG methods were an obvious choice for generating content for release.

Computer-generated puzzle challenges generally tend to be random in layout and to have a "mechanical" feel to their solution, rewarding exhaustive search rather than intuition. In order to incorporate some aspects of human design into our automatically generated content, to increase its value — or even just perceived value — in the players' eyes, we identified the following areas for improvement, based on our experience with similar logic puzzles:

Wall Symmetry Symmetry in the wall layout may increase the impression of intelligent design.

Hint Symmetry Symmetry in the hint layout may increase the impression of intelligent design.

Strategic Depth The more strategies required to solve a challenge, the more interesting it is likely to be for the solver.

Over 100,000 challenges ranging in size from 6×6 to 12×12 were automatically generated, using heuristics to maximise wall symmetry, hint symmetry and strategic depth, then a portion of these were visually inspected and manually tested to arrive at the final 120 levels for release. A separate set of 80 7×7 challenges was randomly selected to populate a user survey, shown in Figure 1.18(b), in which subjects were asked to play a number of challenges and after each one asked whether they thought

it was human- or computer-designed, and how interesting they found it compared to other challenges they'd seen.

We found it interesting that subjects deemed around 50% of the challenges to be human-designed, even though *every single one was computer-designed*. It appears that the mere suggestion that a puzzle might be human-designed can be enough to influence players' perception of it. This experiment did not provide any significant correlation between the players' perception of a challenge and their enjoyment of it, but it demonstrates axes along which procedural methods for puzzle generation may be adjusted to make the resulting output appear more "human" in players' eyes, for future application. For example, Figure 1.19 shows the noticeable difference between random hint placement (left) and symmetrical hint placement (right).

1.3.6 Open World Games

Subversion was a commercial game prototype being developed by Introversion Software[10] which has sinced morphed into their latest release: Prison Architect. The espionage and crime gameplay planned for Subversion took place in a virtual world featuring procedurally generated landscapes and cityscapes produced by a custom 3D generation engine. Moreover, the Subversion engine generated the terrain and cities on the fly in a bespoke way at the start of each game. In this context, we took the opportunity to work with the prototype to test the hypothesis that evolutionary techniques could be employed to customise the city environment in an efficient manner which adds value to the game, i.e., looking at pre-gameplay user-driven adaptation.

In [32], we looked at ways in which the user could change the overall look of the city by specifying some parameters for a fitness function which drove the automatic evolution of a pixel shader [24]. We employed the well-known OpenGL Shading Language (GLSL), which is described in [50], and first abstracted the code for the shaders to a tree representation, where the 3D co-ordinates of a pixel to be rendered are passed through the tree, with the output from the tree being the colour to render the pixel in the user's viewport. The nodes of the tree performed arithmetic manipulations, boolean checks, calculated norms and worked with the pixel's diffuse and specular lighting components. An example tree is presented in Figure 1.21, along with the flattened GLSL version of it which was compiled for the shader, the results of which can also be seen rendered on an example city in Figure 1.21.

The user-specified fitness function for the evolutionary search involved the hue, saturation and luminance of the shader, and the relative importance of these aspects. Given the nature of the proposed application, which involves the player waiting while a pixel shader is evolved at the start of a game, we restricted our experimentation to short sessions, in particular of only ten generations. In a series of 21 sessions, varying over six fitness functions, we showed that, on average, the fitness of the best individual raised to 92.0 from 69.7, which was the fitness of a randomly generated

[10] www.introversion.co.uk

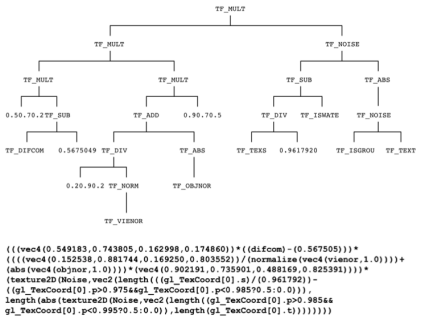

Figure 1.20 An (incomplete) example shader tree and compiled script.

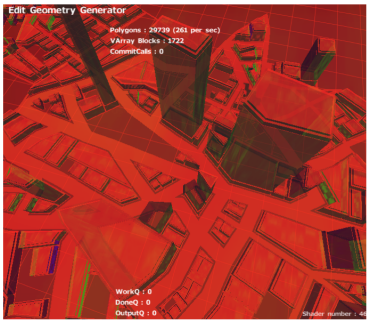

Figure 1.21 The resulting rendering by the pixel shader.

individual. Details of the experimentation and results are given in [32], and in Figure 1.22, we present four evolved pixel shaders.

In our second application with Subversion, we investigated a more interactive approach to evolving game assets, as described in [42]. Individual buildings in the Subversion cities are represented with a plain text markup language that describes how the buildings are built from the ground up as a stack of three dimensional objects. A simple example script, along with the building it generates is given in Figure 1.23(a). The building descriptions are amenable to random generation, crossover and mutation, which enabled us to implement and test a user-driven evolutionary approach to building generation.

Interpreting the building description files as trees, we enabled crossover by first tagging all the branches of two parent trees, so that only branches of the same type could be swapped. We defined the *strength* of a crossover action as the number of branches that were swapped. In addition, we implemented two versions of mutation, namely structural and parametric. With the first of these, a subtree of a given building representation is swapped for a randomly generated subtree, while in the second case, certain numerical parameters in the description are randomly varied. As for crossover, we defined the strength of the mutation as the number of subtrees replaced and the number of parameters varied respectively for the two approaches. Example crossover and mutation actions are depicted in Figure 1.23(b).

In general, during user-driven evolutionary art and design projects, the user chooses artefacts such as pictures or 3D objects based on their phenotypes, i.e., their visual properties, and then the software crosses over and mutates their genotypes, i.e., their underlying data structure and/or the programs that were used to produce them. One issue that often arises in such projects is how satsifying the process feels to the user. In particular, if the children of chosen parents consistently look too similar to the parents, then the user is likely to feel that they are making too slow progress. Conversely, however, if the children look too dissimilar to their parents, then the user is likely to feel that their choices are not really driving the evolution. Hence a middle

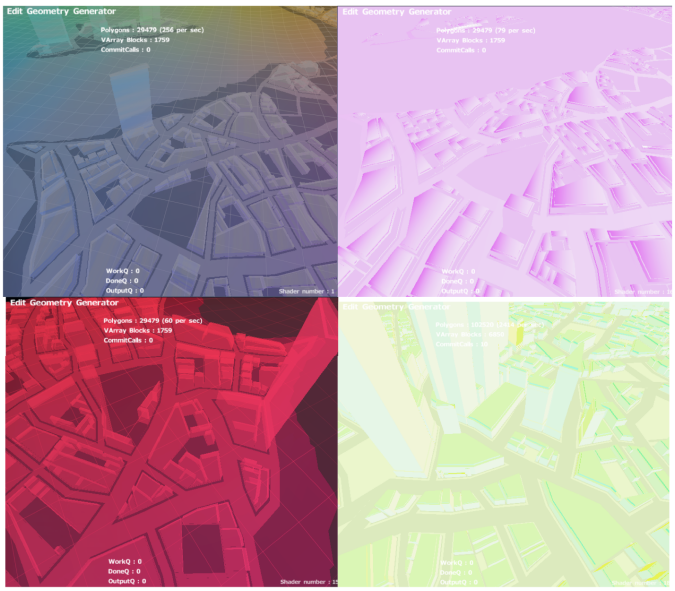

Figure 1.22 Four examples of evolved pixel shaders

ground has to be sought and often initial experimentation is required to fine-tune the evolutionary parameters to find this happy medium.

In our case, we performed initial experimentation to help determine the optimal strength of the crossover and mutation operators when the evolutionary process was driven by the user making phenotype choices. For twenty pairs of parent buildings, we asked 10 participants to comment upon whether each of sixteen generated children were (a) too similar (b) too dissimilar or (c) neither, with respect to their parents. Some of the children were generated via crossover of material from both parents, some were generated via a mutation of a single parent, and some were randomly generated with no reference to either parent, as a control set. Moreover, the strength of the operations were varied across the sixteen children.

In summary, we found the results very encouraging, as evidenced by (i) only 31.9% of the randomly generated control set were deemed satisfactory, i.e., as neither too similar nor dissimilar, whereas 50% of those generated from parents were deemed satisfactory and (ii) it was generally observed that weaker crossover and mutation operators more often produced children rated as too similar than stronger operators, which more often produced too dissimilar children. Of most interest, we found that the weak form of both crossover and mutation operators more often led to satsifactory buildings than the stronger forms. We used these findings to choose the settings for crossover and mutation in a user-driven GUI which we found to be very useful in designing buildings which, when used in cities, gave them a bespoke and

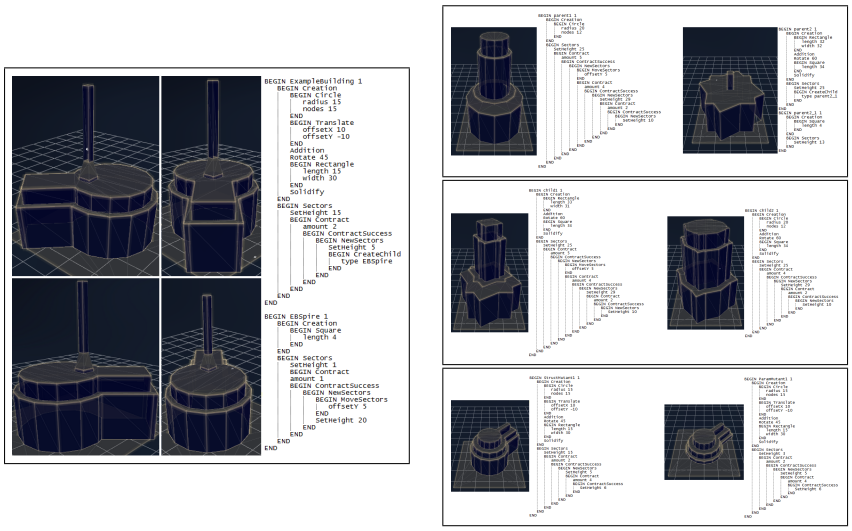

Figure 1.23 (a) An example building, along with the script responsible for its generation. (b)(i) Two example buildings and their Subversion command code descriptions (ii) Two children produced by crossing over the parents above (iii) A structural mutant [left] and a parametric mutant [right].

interesting look and feel. Further details of the experimentation and results are given in [42].

In addition to the two projects described above, we also experimented with evolutionary appraoches to controlling the overall building composition of the cities, e.g., flow of residential (small) buildings into areas of commercial (tall) buildings. We further experimented with traffic flow in the cities, but rather than looking at the usual question of increasing traffic flow, we concentrated on setting up situations for car crashes and other entertainment-based scenarios. In both cases, we found the (unpublished) results very encouraging, further highlighting the potential for evolutionary approaches to both increase/enhance/personalise the game content and to increase user enjoyment in the game.

1.3.7 Summary

The applications described above demonstrate the components of an adaptive system in action.

The *Rogue Trooper* and *The Hunter* results demonstrate that complex, heterogeneous player behaviours, of the kind found in open-world games, can be successfully captured in simple trait models. These models can be learned in an unsupervised way from game log data. They allow meaningful comparisons to made between players in complex gaming environments, which is crucial for adaptation.

The *Pac-Man* results show that player behaviour and relevant game metrics to distinguish playing style can be automatically captured in small-scale action games using linear analysis methods. In particular, the dimensionality reduction property of linear discriminant analysis is very useful in representing player behaviour in very few variables and also automatically weighing metrics by importance. The *Super Mario* results demonstrate the viability of using player traits that were obtained in an initial survey using unsupervised analysis methods to automatically generate a level tailored to the (estimated) skill of a player.

ANGELINA demonstrates adaptation of content to the output of other generative content systems, and how such relationships can be carefully managed to encourage co-operation between the output of such systems, producing content that is generated independently but designed with a shared goal.

The board game applications (LUDI and *Shibumi*) and puzzle application (*Hour Maze*) demonstrate that the measurement of game content, validated by user surveys that gauge the users' playing experience, is sufficient to guide the automated search for interesting new content. The *Shibumi* project, in particular, will comprise a complete system for the automated generation and adaptation of bespoke board games, according to the user's playing style and preferences, when completed.

The *Subversion* application further demonstrates the benefit of a user-driven evolutionary process, based on the users' experience, for generating game-environment — as opposed to game-play — content.

1.4 Conclusions

In the last five years, we have worked directly with five commercial video games companies: Introversion Software, Rebellion Developments, Emote Games, Lionhead Studios and Nestorgames. Some of the results of these interactions have been described in this chapter, and in each case, there has been overall (long-term) goals of (i) helping industry practitioners to build better, adaptive games, and (ii) studying the potential for creative software to generate game content and even entire games, from a Computational Creativity perspective. As a result of the individual projects presented here, our main contribution has been to propose a cycle of generation, measurement and adaptation within an overall methodology for adaptive game design in a context of procedural content generation, as detailed in section 1.2.

From the perspective of designing commercial games that are able to adapt to personalise the gaming experience, there are major obstacles in each part of the cycle. In particular, simply recording gameplay and sensor data about a player may slow down the game so much that it is impractical. If it is possible to record and massage such data into a usable form, then reliably estimating the user experience from this data is very difficult indeed and will require massive amounts of user play-testing and the machine learning of classifiers to be used in-game. Building a game which can alter itself at run-time in the various ways necessary for the alterations to be perceivable, not disorientating and to have the potential to improve matters is a serious engineering challenge. Finally, putting all these aspects together into an automated

game director able to alter the game at run-time in such a way that there is a good chance of improving user experience is a serious research problem for the industry. The individual projects described in section 1.3 each addresses one or more of these difficulties. Moreover, they have helped us to flesh out the benefits of employing the cyclic methodology via experimental results arising from the study of both commercial and experimental games.

In particular, with respect to generative aspects, we have shown that board game rule sets can be automatically developed with commercial success, and that other game assets, such as buildings, levels and pixel shaders can be similarly generated. Moreover, we have shown that entire platformer games can be evolved not only to be entertaining through gameplay, but also to reflect current issues expressed in newspaper articles. With respect to measurement aspects, we have performed extensive analysis of data arising from the playing of games, talking about the experience in interviews, and messaging other players through social networking. This has enabled us to pioneer new ways to capture low-level gameplay data and high-level user experience data, and to show the value of statistical techniques such as linear discriminant analysis for summarising important aspects of how people play games. With respect to adaptive aspects, we have shown how simple games such as *Snakeotron* can adapt as players progress, and how levels for *Super Mario* can be built in response to user data. We have further shown how automatically generated puzzles can be given aspects of a human touch, and the potential for players to evolve the look and feel of a game environment such as the cityscape in *Subversion* before they play the game.

From the perspective of Computational Creativity, we have learned a great deal about the potential for software to be creative in the game design process. In particular, we have looked at how web resources such as newspaper articles, multimedia assets and social network data can drive generative processes, often leading to interesting and surprising results. We have also looked at how individual generative processes can be integrated so that the whole is more than a sum of the parts, and how software can add value to its creations by providing information about how/why it operated and framing its work in various cultural contexts, such as news reporting. We have studied how best to enable users to guide evolutionary processes for content creation, so that they feel satisfied with the level of progress they are making. We have also studied responses from the game playing public to information about the computational genesis of games and game content. Interestingly, we have noticed a lesser effect of so-called *carbon fascism* (i.e., default prejudice against the intelligence/creativity of software) in gaming circles than in other artistic cultures. From a broader AI perspective, we note that gameplay data is becoming more available for machine learning applications, which is an exciting prospect — the availability of such data in other domains such as bioinformatics has revolutionised those areas.

Commercial games with mild aspects of adaptation, such as *Left4Dead*, are already making an impact. Moreover, players now demand huge game worlds, large numbers of non-player characters and other players to interact with, and to be regularly given new missions. This naturally increases their impression of having a bespoke gaming experience each time they play, which, in turn, keeps their interest in the game for longer periods. An interesting question for the future is whether

players want more direct personalisation of their experience, and if so, how to bring this about. We believe that the investigations presented here, developed with academic and industrial collaboration, is indicative of the kind of fundamental research required to bring about a new wave of personalised games which adapt to individual players. The majority of the work described above is very much ongoing, and we intend to find new and interesting projects which raise and answer new questions in adaptive gaming and Computational Creativity research, in order to contribute to this exciting field in the future.

Acknowledgements

We would like to thank all the games industry professionals and academic researchers who have worked with us on games projects of all types. We would like to particularly thank Paul Cairns, Paul Miller, Daniel Ramirez-Cano, Néstor Romeral Andrés, Stephen Tavener, Andrew Howlett, Andrew Lim and Andrew Martin, whose work contributed directly to the projects described above. This work has been supported by EPSRC grants TS/G002835, EP/I001964, EP/J004049/1 and TS/G002886 and TSB grant AL318J.

REFERENCES

1. Ingo Althöfer. Computer-Aided Game Inventing. Technical report, Friedrich-Schiller Univ., Faculty Math. Comp. Sci., Jena http://www.minet.uni-jena.de/preprints/althoefer_03/CAGI.pdf, 2003.
2. Mike Ambinder. Biofeedback in gameplay: How Valve measure physiology to enhance gaming experience. Online slides: http://www.valvesoftware.com/publications/2011/ValveBiofeedback-Ambinder.pdf, 2011.
3. Stylianos Asteriadis, Noor Shaker, Kostas Karpouzis, and Georgios Yannakakis. Towards player's affective and behavioral visual cues as drives to game adaptation. In *Proceedings of Workshop on Multimodal Corpora 2012*, pages 6–9, 2012.
4. Richard Bartle. Hearts, clubs, diamonds, spades: Players who suit MUDs. *Journal of MUD Research*, 1(1), 1996.
5. Chris Bateman, Rebecca Lowenhaupt, and Lennart E. Nacke. Player typology in theory and practice. In *Think Design Play: International Conference of the Digital Games Research Association (DIGRA)*, Hilversum, the Netherlands, September 2011.
6. Robin Baumgarten. Towards automatic player behaviour characterisation using multi-class LDA. In *Proc. AISB Symp. on AI & Games*, pages 63–66, 2010.
7. Michael Booth. The AI systems of Left 4 Dead. Online slides: http://www.valvesoftware.com/publications/2009/ai_systems_of_l4d_mike_booth.pdf, 2009.

8. Ingwer Borg and Patrick Groenen. *Modern Multidimensional Scaling: Theory and Applications*. Springer, 2nd edition, 2005.
9. Cameron Browne. *Automatic Generation and Evaluation of Recombination Games*. Ph.d. dissertation, QUT, Brisbane, 2008.
10. Cameron Browne. *Evolutionary Game Design*. Springer, Berlin, 2011.
11. Cameron Browne. Elegance in Game Design. *IEEE Trans. on Computational Intelligence & AI in Games*, 4, 2012.
12. Cameron Browne. Evolutionary game design: 2012 "Humies" winner. *SIGEVO Newsletter*, 6(2), 2012.
13. Cameron Browne. Metrics for better puzzles. In Magy Seif El-Nasr, Anders Drachen, and Alessandro Canossa, editors, *Game Analytics: Maximizing the Value of Player Data*. Springer, New York, 2012.
14. Cameron Browne and Simon Colton. Computational creativity in a closed game system. In *Proc. Comput. Intell. Games (CIG)*, 2012.
15. Cameron Browne, Edward Powley, Daniel Whitehouse, Simon Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A Survey of Monte Carlo Tree Search Methods. *IEEE Trans. on Computational Intelligence & AI in Games*, 4:1:1–43, 2012.
16. Cameron Browne and Néstor Romeral Andrés. *Shibumi Rule Book*. Lulu, Raleigh, 2012.
17. Simon Colton and Geraint A. Wiggins. Computational creativity: The final frontier. In *Proceedings of the European Conference on Artificial Intelligence*, 2012.
18. Michael Cook and Simon Colton. Initial results from co-operative co-evolution for automated platformer design. In *Volume 7248 of Applications of Evolutionary Computation*, 2012.
19. Michael Cook, Simon Colton, and Alison Pease. Aesthetic considerations for automated platformer design. In *Proceedings of the 8th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, Forthcoming, 2012.
20. Benjamin Cowley. *Player Profiling and Modelling in Computer and Video Games*. PhD thesis, University of Ulster, Belfast, UK, 2009.
21. Chris Crawford. *The Art of Computer Game Design*. McGraw Hill, Berkeley, 1984.
22. Ethan Dereszynski, Jesse Hostetler, Alan Fern, Tom Dietterich Thao-Trang Hoang, and Mark Udarbe. Learning probabilistic behavior models in real-time strategy games. In *Proceedings of the Seventh AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2012.
23. Anders Drachen, Alessandro Canossa, and Georgios Yannakakis. Player modeling using self-organization in Tomb Raider: Underworld. In *Proc. IEEE Symp. on Comp. Intelligence & Games (CIG)*, pages 1–8, 2009.
24. Wolfgang Engel. *Programming Vertex and Pixel Shaders*. Charles River Media, 2009.
25. Jeremy Gow, Robin Baumgarten, Paul Cairns, Simon Colton, and Paul Miller. Unsupervised modelling of player style with LDA. *IEEE Trans. on Computational Intelligence & AI in Games*, 4(3), 2012.
26. Jeremy Gow, Simon Colton, Paul Cairns, and Paul Miller. Mining rules from player experience and activity data. In *Proceedings of the Eighth Annual AAAI Conference*

on Artificial Intelligence and Interactive Digital Entertainment. The AAAI Press, 2012. Forthcoming.

27. Eun Y. Ha, Jonathan P. Rowe, Bradford W. Mott, and James C. Lester. Goal recognition with markov logic networks for player-adaptive games. In *Proceedings of the Seventh AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2012.
28. Erin J. Hastings, Ratan K. Guha, and Kenneth O. Stanley. Evolving content in the galactic arms race video game. In *Proc. IEEE Symp. on Comp. Intelligence & Games (CIG)*, 2009.
29. Mark Hendrikx, Sebastian Meijer, Joeri van der Velden, and Alexandru Iosup. Procedural content generation for games: A survey. *ACM Trans. Multimed. Comput. Commun. Applic.*, 2011.
30. Hiroshi Higashida. Machine-made puzzles and hand-made puzzles. In R. Nakatsu *et al.*, editor, *IFIP Adv. Inform. Commun. Technol.*, volume 333, pages 214–222, 2010.
31. Victor Hom and John Marks. Automatic design of balanced board games. In *Proc. 3rd Artif. Intell. Interact. Digital Entert. Conf.*, pages 25–30, 2007.
32. Andy Howlett, Simon Colton, and Cameron Browne. Evolving pixel shaders for the prototype video game subversion. In *Proceedings of the AISB symposium on AI and Games*, 2010.
33. Robin Hunicke. The case for dynamic difficulty adjustment in games. In *Proceedings of the International Conference on Advances in Computer Entertainment Technology, ACE 2005, Valencia, Spain, June 15-15, 2005*, pages 429–433. ACM, 2005.
34. Hiroyuki Iida, Kazutoshi Takahara, Jun Nagashima, Yoichiro Kajihara, and Tsuyoshi Hashimoto. An application of game-refinement theory to mah jong. In *Proc. ICEC'04, LNCS 3166*, pages 333–338, 2004.
35. Martin Jennings-Teats, Gillian Smith, and Noah Wardrip-Fruin. Polymorph: dynamic difficulty adjustment through level generation. In *Proceedings of the 2010 Workshop on Procedural Content Generation in Games, PCGames 2010*, New York, 2010. ACM.
36. Nobuhiko Kanamoto. A well-made *Sudoku* is a pleasure to solve, 2001. Available at `http://www.nikoli.co.jp/en/puzzles/sudoku/hand_made_sudoku.htm`.
37. Scott Kim. What is a puzzle?, 2008. Available at `http://scottkim.com/thinkinggames/whatisapuzzle/index.html`.
38. John Koza. *Genetic Programming*. MIT Press, Cambridge, 1992.
39. Nicole Lazzaro. Why we play: Affect and the fun of games. In *In The Human- Computer Interaction Handbook: Fundamentals, Evolving Technologies, and Emerging Applications*, pages 679–700. Lawrence Eribaum, New York, 2003.
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. ACM, 1980.
41. 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.
42. Andrew Martin, Andrew Lim, Simon Colton, and Cameron Browne. Evolving 3d buildings for the prototype video game subversion. In *Proceedings of the EvoGames Workshop*, 2010.

43. Geoffrey. J. McLachlan. *Discriminant Analysis and Statistical Pattern Recognition*. Wiley Interscience, 2004.
44. Michelle McPartland and Marcus Gallagher. Reinforcement learning in first person shooter games. *IEEE Trans. on Computational Intelligence & AI in Games*, 3(1), 2011.
45. Gwaredd Mountain. Psychology profiling in Silent Hill: Shattered Memories. Invited talk at Paris Game/AI Conference, 2010. Available from `http://gameaiconf.com/?p=141`.
46. Christopher Pedersen, Julian Togelius, and Georgios Yannakakis. Modeling player experience for content creation. *IEEE Trans. on Computational Intelligence & AI in Games*, 2(1):54–67, 2010.
47. Mitchell Potter and Kenneth De Jong. A cooperative coevolutionary approach to function optimization. In Yuval Davidor, Hans-Paul Schwefel, and Reinhard Männer, editors, *Parallel Problem Solving from Nature — PPSN III*, volume 866 of *Lecture Notes in Computer Science*, pages 249–257. Springer Berlin / Heidelberg, 1994.
48. Andrew K. Przybylski, C. Scott Rigby, and Richard M. Ryan. A motivational model of video game engagement. *Review of General Psychology*, 14(2):154, 2010.
49. Daniel Ramirez-Cano, Simon Colton, and Robin Baumgarten. Player classification using a meta-clustering approach. In *Proceedings of the 3rd International Conference on Computer Games, Multimedia and Allied Technology, CGAT 2010, Singapore, April 2010*, 2010.
50. Randi Rost, Bill Licea-Kane, Dan Ginsburg, John Kessenich, Barthold Lichtenbelt, Hugh Malon, and Mike Weiblen. *OpenGL Shading Language, 3rd Edition*. Addison Wesley, 2009.
51. R. Wayne Schmittberger. *New Rules for Classic Games*. John Wiley & Sons, New York, 1992.
52. Noor Shaker, Julian Togelius, Georgios N. Yannakakis, Ben Weber, Tomoyuki Shimizu, Tomonori Hashiyama, Nathan Sorenson, Philippe Pasquier, Peter Mawhorter, Glen Takahashi, Giliian Smith, and Robin Baumgarten. The 2010 mario ai championship: Level generation track. *IEEE Trans. on Computational Intelligence & AI in Games*, 3(4):332–347, 2011.
53. Noor Shaker, Georgios Yannakakis, Julian Togelius, Miguel Nicolau, and Michael O'Neill. Evolving personalized content for Super Mario Bros using grammatical evolution. In *Proceedings of the Seventh AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2012.
54. Pieter Spronck, Marc Ponsen, Ida Sprinkhuizen-Kuyper, and Eric Postma. Adaptive game ai with dynamic scripting. *Machine Learning*, 63(3):217–248, 2006.
55. Penelope Sweetser and Peta Wyeth. Gameflow: a model for evaluating player enjoyment in games. *Computers in Entertainment (CIE)*, 3(3):3–3, 2005.
56. Ruck Thawonmas and Keita Iizuka. Visualization of online-game players based on their action behaviors. *International Journal of Computer Games Technology*, 2008.
57. Mark J. Thompson. Defining the Abstract. *The Games Journal*, 2000.
58. Julian Togelius, Georgios Yannakakis, Kenneth O. Stanley, and Cameron Browne. Search-based procedural content generation: A taxonomy and survey. *IEEE Trans. on Computational Intelligence & AI in Games*, 3(3):172–186, 2011.

59. Gregory Trefay. *Casual Game Design: Designing Play for the Gamer in All of Us*. Morgan Kaufmann, Burlington, 2010.

60. Giel van Lankveld, Pieter Spronck, Jaap van den Herik, and Arnoud Arntz. Games as personality profiling tools. In *Proc. IEEE Symp. on Comp. Intelligence & Games (CIG)*, pages 197–202, 2011.

61. Georgios Yannakakis, M. Maragoudakis, and John Hallam. Preference learning for cognitive modeling: a case study on entertainment preferences. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 39(6):1165–1175, 2009.

62. Georgios N. Yannakakis. How to model and augment player satisfaction: A review. In *Proceedings of the 1st Workshop on Child, Computer and Interaction, Chania, Crete, ACM Press*, 2008.

63. Georgios N. Yannakakis and Julian Togelius. Experience-driven procedural content generation. *IEEE Trans. on Affective Computing*, 2(3):147–161, 2011.

64. Nick Yee, Nicolas Ducheneaut, Les Nelson, and Peter Likarish. Introverted elves & conscientious gnomes: The expression of personality in world of warcraft. In *Proc. Int. Conf. on Human Factors in Computing Systems (CHI)*, pages 753–762, 2011.

65. Alexander E. Zook and Mark O. Riedl. A temporal data-driven player model for dynamic difficulty adjustment. In *Proceedings of the Seventh AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2012.