Gameplay semantics for the adaptive generation of game worlds

Proefschrift

ter verkrijging van de graad doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof. ir. K.C.A.M. Luyben,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op dinsdag 16 september 2014 om 10:00 uur

door

Ricardo LOPES

Mestre em Engenharia Informática,
Instituto Superior Técnico, Portugal
geboren te Lissabon, Portugal.

Dit proefschrift is goedgekeurd door de promotor:
Prof. dr. ir. E. Eisemann

Copromotor:
Dr. ir. R. Bidarra

FCT Fundação para a Ciência e a Tecnologia
MINISTÉRIO DA EDUCAÇÃO E CIÊNCIA

This research has been funded by Fundação para a Ciência e Tecnologia, the Portuguese national funding agency for science, research and technology.

ISBN 978-90-8891-968-8
©2014, Ricardo Lopes, Delft, All rights reserved.
Thesis published by: Uitgeverij BOXPress, ’s-Hertogenbosch

127168

Contents

1 Introduction

Video games are a mainstream form of entertainment, increasingly common in a vast number of households. In the last forty years, the rapid advancements in video game technology have followed from the cultural and social ever-growing acceptance of this form of entertainment. People of all demographics regularly play with their computers, TVs or portable devices [ESA 13], immersing themselves in ever-richer and deeper virtual worlds.

However, many players and researchers still feel that games could somehow be even more fun and engaging. This can be apparent, for example, from one statistical fact: the low average rate of players who complete games they bought (reported from 10% [CNN 11] to 20 - 25% [IGN US 11]). In fact, this data has been analyzed to conclude that most commercial video games do not include engaging single-player campaigns [Game Front 12].

We identify one of the possible reasons for such lack of engagement: the *rigidity* of games. When most commercial games are shipped, their gameplay has typically been pre-scripted. All game components, *e.g.* characters, rules, narratives and environments, are created during development, mostly as pre-determined artifacts with which a player will interact. In an attempt to account for more *flexible* gameplay, video games often include minor variations that depend on self-profiling. The most common example is the customization of a game's difficulty, where players choose to experiment different sets of game conditions, with varying degrees of challenge. However, this discrete approach is still somehow limited since it is typically constrained by the low-resolution of the available game options (*e.g.* self-classification as beginner vs. expert).

Such lack of flexibility can significantly hinder player engagement. With rigid game content and its fixed discrete variations, player engagement is dependent on how good these game components were designed to fit that player. Furthermore, game outcomes can be more easily anticipated by players, since all possible interactions are bounded by such components. Even worse, if players can predict certain outcomes, on a regular basis, they can repeatedly exploit those predictions to progress, resulting in a less natural game experience. Such inflexibility can also affect some of the replay value of such games. Content-wise, very little new or different can be discovered when replaying previously explored games.

Furthermore, video games could do better in attracting and retaining a wider audience of players. Being rigid, games are typically designed with a certain player type in mind. This leads to player specialization, since new players must learn how to become that player type or else be left out. Additionally, the discrete self-profiling discussed above implies that such games might fail in appealing to players who do not know how to profile themselves or who do not identify themselves with any of the available classifications.

1.1 Adaptive games

Several game features have already been developed to help account for these issues. An example is the dynamic adjustment of a game's difficulty level to match the measured skill of the player. The generation of specific game events, adjusted to the pace and behavior of a player, and of linear game levels, adapted to the measured emotional states of players, are other examples. Games which feature such automatic adjustments are termed as *adaptive*.

An adaptive game automatically customizes itself to better fit an individual player. Its components are no longer pre-determined, being able to dynamically change themselves to fit player-centered requirements (*e.g.* an easier level, generated on-the-fly). These types of games can cater the gaming experience to the individual user, by being more responsive than normal games to different player types and their individual needs [Charles 05, Gilleade 04, Magerko 08a]. As such, they can be played in a more dynamic and flexible fashion, potentially engaging many more players.

Methods to support adaptive games already exist. The standard approach for an adaptive game typically includes two components: a *player model* and an *adaptation mechanism*. A player model is created by an algorithm that assesses and predicts player behavior, by analyzing gameplay data and inferring a mathematical model of players' actions, preferences or style [Houlette 04]. In adaptive games, this player model steers some adaptation mechanism by requiring specific game changes to fit the model state. One of the possible adaptation mechanisms is procedural content generation (PCG), an umbrella term for algorithms that can automatically generate a specific type of content [Smelik 11a], with limited or indirect user input. Content typically refers to most of what is contained in a game (*e.g.* a game level, a story), except for the game engine itself or non-player characters (NPC) behavior [Togelius 11].

1.2 Problem statement

Although adaptive games are a possible solution for the player engagement issues mentioned before, in practice they are rare. The technology and the development techniques to support adaptive games are not easily accessible to game designers, less technically savvy than programmers. In turn, this creates obstacles when authoring adaptivity, *i.e.* supplying designers with the opportunities to control how the player model and adaptation mechanism should respond to individual gameplay. Additionally, most adaptive game techniques are strongly *ad-hoc*, most of the times developed for a specific case, without further application beyond it. Such dedicated approaches are also limited in how and which components they are able to adapt (*e.g.* only adjusting spawn points for enemy NPCs).

Apart from some noteworthy examples (see Chapter 2), the development of adaptive games is in its infancy, especially when considering PCG as the adaptation mechanism. Current PCG research shows that generating content can already be considered very broad (*e.g.* levels, maps, textures, stories, events), produce highly diversified artifacts and be effectively controllable. These qualities can have a high impact on adaptive games, by providing more ways to adapt better. Therefore, it is important to focus on the development of PCG-based adaptive games.

This dissertation's goal is to contribute towards the development of adaptive games, by addressing all the issues described before with a specific focus on PCG-based adaptive games. We contribute to: (i) empower designers to author and control adaptivity in games (since their knowledge on gameplay is too rich to not be used), (ii) supply generic technology, applicable across different game genres, and (iii) support more features on what can be adapted and to what purpose. We focus these aims on the generation of complex game worlds, the target of our PCG-based adaptive games. Generating game worlds, dependent on a player model, is a far reaching adaptation mechanism but it remains an open research topic [Lopes 11b].

Solving these aims can considerably profit from recent achievements in semantic modeling, where the control, features and generic nature of game world generation have been successfully improved [Tutenel 12]. Game world semantics is all information on a game world and its objects beyond their mere visual representation. For example, the weight or capacity a box possesses, in a virtual game world. Semantics can act as the knowledge base which constrains and steers the procedural generation of complex artifacts as, for example, buildings [Tutenel 11a].

We propose the use of semantics to adaptively generate game worlds. We extend the current semantics model with the notion of *gameplay semantics*, defined as:

> the knowledge on the *gameplay* meaning and value of a game world and its objects.

For example, the *fun* the same box is able to provide to a certain known *player style*. Our proposal is that gameplay semantics acts as the knowledge which *glues* together player models and the appropriate game adaptation mechanism, steering the latter. In Chapter 3, this definition of gameplay semantics is fully realized into a formal framework.

1.3 Research question

All of this lead us to the following main research question:

- *How can gameplay semantics improve the adaptive generation of game worlds?*

To answer our research question, we will answer the following key questions:

1. How can gameplay semantics steer the adaptive generation of game worlds?
2. Which game world features can be generated from gameplay semantics?
3. How can game designers use gameplay semantics to author adaptive game world generation?
4. Which games, genres, player modeling and PCG methods can gameplay semantics apply to?

1.4 Methodology

The methodology we followed to answer these questions was divided into three stages. First, we created a novel specification model of gameplay semantics, by extending current semantic modeling methods. A generic generation method was included in the specification model to allow easy integration into procedural content generators. Second, we applied this approach to several case studies, with games and methods of different characteristics. Finally, we assessed the effectiveness and controllability within these case studies with players and game designers.

1.5 Outline of contributions

Our contributions emerge from answering the research questions above and are presented in this dissertation, where each chapter corresponds to a published article. Furthermore, each chapter is backed-up by a running software, typically composed of: a game, a generator and a player modeling method.

The sequence of chapters in this dissertation illustrates the path to answering our research questions. We start by validating the importance of adaptive game worlds and of PCG as a method to support them, as well as identifying the research opportunities for semantics to improve both (Chapter 2). Next, we elaborate what is gameplay semantics and how it can be used to steer adaptive game world generation, by proposing a semantic model and a generation framework for adaptive game worlds (Chapter 3). In our first case study, we focus on determining how game designers can control adaptive game world generation, while investigating the generation of specific game world layout features (Chapter 4). Further generable game world features are investigated, with a focus on global (Chapter 5) and specialized (Chapter 6) properties of game worlds. Throughout Chapters 3, 4 and 5 not only we investigate such features, but also what games, genres, player modeling and PCG methods can benefit from gameplay semantics.

Each chapter encapsulates broader contributions to this field. Chapter 2 surveys the current state of adaptive games research and technology, discussing the main unexplored research opportunities. In particular, it concludes that PCG and semantic modeling can powerfully combine to support the development of adaptive games. This survey was published in IEEE Transactions on Computational Intelligence and AI in Games: *Adaptivity challenges in games and simulations: a survey* [Lopes 11b].

Chapter 3 proposes a generation framework aimed at adaptively creating content for complex and immersive game worlds. It introduces the core definition of gameplay semantics and describes how it can be integrated within an adaptive game (game, generator and player model), in a generic way. This work was published in the eighth International Conference on Advances in Computer Entertainment technology: *A semantic generation framework for enabling adaptive game worlds* [Lopes 11a].

Chapter 4 discusses how gameplay semantics can be used to procedurally generate adaptive game worlds, when considering off-line generation of 3D racing stunt arenas, using an heuristic-based player model. Results show that semantics can be effectively used by designers to *control* PCG and make it fit personal gameplay. Additionally, we discuss how gameplay semantics can generate emergent content, *i.e.* beyond the designer pre-specification. This work on adaptive off-line game worlds was published in the third workshop on Procedural Content Generation in games: *Using gameplay semantics to procedurally generate player-matching game worlds* [Lopes 12].

Chapter 5 further investigates semantics-based adaptivity and respective designer-centered control, but considering on-line generation of 2D platform levels for mobile games. It also shows how gameplay semantics can be used effectively in conjunction with other forms of adaptivity, based not on semantics or player models but on real-world time constraints placed on the player. Performed user studies show that this approach successfully accommodates different player types, adapting and improving gameplay. This work was published in the fourth workshop on Procedural Content Generation in games: *Mobile adaptive procedural content generation* [Lopes 13b] and in the eighth International Conference on the Foundations of Digital Games: *Gameplay semantics for authoring adaptivity in mobile games* [Lopes 13a].

Chapter 6 discusses how semantics can be used for more complex on-line and constraint-based generation of 3D maze-like levels. Specifically, this chapter shows how semantics can contribute towards enabling designers to author adaptivity in game world generation, in a more *expressive* and *specific* fashion than before. User studies matching both designers and players showed that gameplay semantics can provide game designers with a rich expressive range to convey specific adaptive gameplay experiences to its players. This work has been submitted for journal publication.

The final chapter 7 discusses our research results and draws conclusions and future recommendations from them.

In addition to the above, throughout this project, we supervised and gave substantial contributions to research work related to game level and game world generation. This resulted in the following co-authored publications:

- *A constrained growth method for procedural floor plan generation* [Lopes 10], in GAME-ON - Simulation and AI in Games Conference 2010.
- *Generating consistent buildings: a semantic approach for integrating procedural techniques* [Tutenel 11a], in IEEE Transactions on Computational Intelligence and AI in Games.
- *Designing procedurally generated levels* [van der Linden 13a], in the second AAAI Workshop on Artificial Intelligence in the Game Design Process.
- *Procedural dungeon generation* [van der Linden 13b], in IEEE Transactions on Computational Intelligence and AI in Games.
- *A generic method for classification of player behavior* [Etheredge 13], in the second AAAI Workshop on Artificial Intelligence in the Game Design Process.

2 Adaptivity challenges in games and simulations

In computer games and simulations, content is often rather static and rigid. As a result, its pre-scripted nature can lead to predictable and impersonal gameplay, while alienating unconventional players. Adaptivity in games has therefore been recently proposed to overcome these shortcomings and make games more challenging and appealing.

In this chapter we survey present research on game adaptivity, identifying and discussing the main challenges, and pointing out some of the most promising directions ahead. We first survey the *purposes* of adaptivity, as the principles that could steer an adaptation and generation engine. From this perspective, we proceed to thoroughly discuss adaptivity's *features* and *methods*.

We conclude that, among other methods, procedural content generation and semantic modeling can powerfully combine to create off-line customized content and on-line adjustments to game worlds. These and other promising methods, deserving ample research efforts, can therefore be expected to significantly contribute towards making games and simulations even more unpredictable, effective and fun.

Publication notice: *The scientific content of this chapter was published in the IEEE Transactions on Computational Intelligence and AI in Games [Lopes 11b]*

2.1 Introduction

Typically, when most commercial games are shipped, their gameplay has been prescripted. The same happens with simulations, which generally use game technology to emulate reality and training conditions. In both cases, game content, rules, narratives and environments are created during the development phase, mostly as *static* elements with which a *dynamic* player will interact. Designing such predefined content is standard because it allows games and simulations to remain robust, testable and controllable. As a result of such rigidity, game outcomes can be more easily anticipated by players, since all possible interactions are bounded by such static elements. Even worse, if players can predict certain outcomes, their progress can be often achieved by repeatedly exploiting a successful strategy.

In an attempt to account for player individuality, games often include minor variations that depend on players profiling themselves. For example, by customizing the difficulty level or choosing time constraints, players are classifying themselves as one of the available pre-defined low-resolution stereotypes, *e.g.* beginners or experts. However, this discrete approach implies that such games might fail in appealing to players who do not know how to profile themselves or who do not identify themselves with any of the available classifications.

Static game content and its pre-defined variations, based on low-resolution profiles, all lead to games and simulations that can be played in an impersonal, predictable and inflexible fashion and that can fail to appeal to broader audiences.

For games with purposes other than entertainment, such as serious games and simulations, these problems can become more acute. Players who need to capture or practice certain skills, all have different learning abilities and training needs. However, serious games and simulations typically do not take such a high-resolution player individuality into account. Current *ad-hoc* and stereotyped training conditions can induce players to mostly perform the same exercises in the same conditions, adding little value to the learning process. This lack of player individuality can also affect the replay value of such games, since nothing new or different can be experienced in consecutive game sessions.

To solve the above shortcomings, many researchers agree that serious games and simulations have to become more challenging, unpredictable and player-centric, to be fully embraced as an effective way of knowledge transfer [Aldrich 02, Blackman 05]. Several other researchers claim that entertainment games should also address these issues, by catering the gaming experience to the individual user, being more responsive to different player types and their individual needs, and adapting themselves to better fit the players [Charles 05, Gilleade 04, Magerko 08a].

Player-centered game adaptivity can help accomplishing the above goals. Dynamically adjusting game elements, according to the individual performance of the player (*i.e.* personal gameplay), can contribute to make the game experience more unique and personal. Consider the example of a driving simulation where a player is monitored as speeding more than desired. An adaptive game could adjust the city environment to discourage this behavior. Examples could be either increasing the number of speed bumps, traffic radars or police patrols or generating more curved roads, stoplights or crosswalks, depending on the player's experience and personality.

In this chapter, we survey the present state of adaptivity in games and simulations, identify the main challenges ahead and discuss possible research directions to tackle them. Fig. 2.1 lays out the architectural principles that drive research on adaptive games. These principles were already latent in the preliminary proposals of Houlette [Houlette 04], Charles [Charles 05] and Magerko [Magerko 08a], as well as in the vast majority of the research that followed. In essence, game logs, recording the player performance, are used to create models of player actions, preferences or personality. Given a game state, these models assess and predict the players desired experience for the next game state. Models for the player experience and performance are then used to steer an adaptation and generation engine, which adjusts the appropriate game components to better fit both.

This survey discusses game adaptivity research from an adaptation and generation perspective. We strongly focus on how (*methods*) and to what (*features*) adaptation and generation engines can or could adapt. By investigating the input and output of an adaptation and generation engine, we are able to formulate our key research questions and reflect on its answers (see Chapter 1): how can adaptive generative methods be steered (and authored) and which features are important to generate in an adaptive game.

Fig. 2.1 illustrates that steering adaptation, *i.e.* controlling it, is strongly related

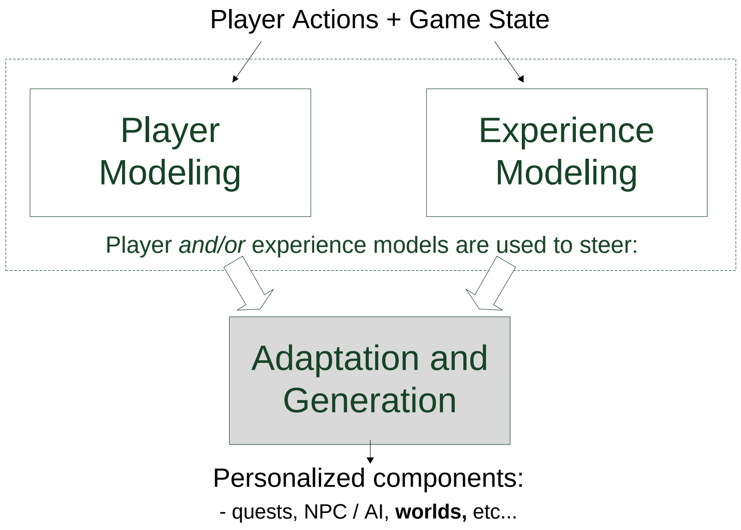

Player Actions + Game State
Player Modeling
Experience Modeling
Player and/or experience models are used to steer:
Adaptation and Generation
Personalized components:
- quests, NPC / AI, worlds, etc...

Figure 2.1: Overview of game adaptivity architectural principles: player and experience modeling steer adaptation and generation of personalized game components

with what is captured in player and experience models. Such models: (i) represent the *purposes* of adaptivity, *i.e.* its objectives (*e.g.* player skill is modeled to adapt the challenge of the game), and (ii) can, ultimately, serve as input and steer adaptation and generation. We discuss these *purposes* from a generic perspective, independent of player modeling and player experience prediction. An in-depth analysis of their implementation through player modeling and experience prediction techniques will therefore not be considered here. Player modeling principles have been already discussed by several researchers [Houlette 04, Beal 02, Thue 07], and experience prediction has been recently surveyed by Yannakakis and Togelius [Yannakakis 11].

This chapter is structured as follows: in Section 2.2 we look at the *purposes* for adapting, by analyzing what is being presently done in steering adaptivity in games and simulations. In Section 2.3 we focus on adaptivity *features*, surveying standard adaptive game components (*e.g.* non playing characters). In Sections 2.4 and 2.5, we survey and discuss, respectively, off-line and on-line *methods* which can be used to adapt game content, before our final conclusions in Section 2.6.

2.2 Steering Adaptivity

In games and simulations, adaptivity can be used to better suit the game to a dynamic element, for example, the skills of a player, the size of a team or the physical environment in which the game is played. As highlighted in Section 2.1, this chapter focuses on player-centered adaptivity, *i.e.* adjustments which improve the individual player experience. For adaptivity to achieve this goal, it needs to be steered by some purpose that game designers can identify, measure and influence. As such, these purposes are especially important since they are both the motivation and the potential interface for game designers to author adaptive gameplay.

Knowledge on this steering purpose will determine how adaptation algorithms decide what, when and how to adjust. For taking this decision, algorithms should identify: (i) what triggers the need for adjustments and (ii) what should be adjusted. For example, if difficulty adjustment is the steering purpose, an adaptive game needs to recognize that consecutive failures may be a sign of high difficulty. It also needs to know concrete in-game ways of affecting the difficulty level. Understanding and choosing what to use to steer adaptivity is both an essential step and a major challenge, required to ensure that game adjustments induce the personalized player experience, on the desired way (*e.g.* adjusting difficulty in the previous example).

Player modeling is the traditional approach to capture and process the necessary information to steer adaptivity. With player models, gameplay information and metrics are processed to create knowledge about the behavior of the player. Player modeling has recently been broadly surveyed in [Smith 11].

In this section, we survey adaptivity's purposes, *i.e.* the generic principles that support player modeling and experience prediction and steer game adaptation methods.

2.2.1 Entertainment games

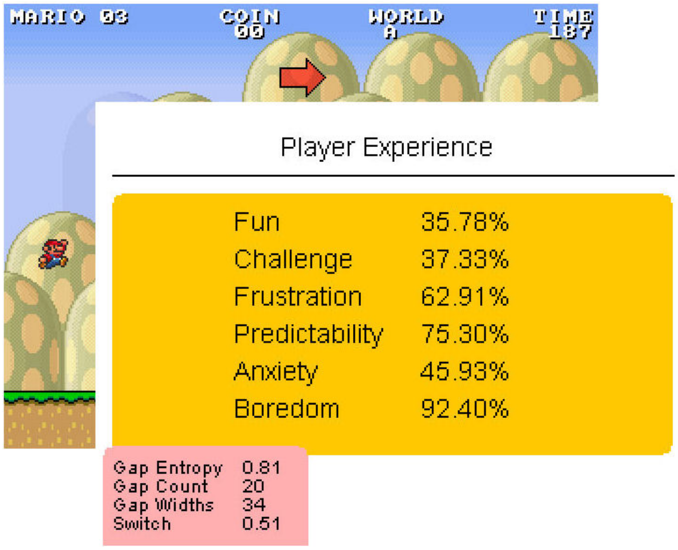
For entertainment games, fun is the fundamental purpose. There are many different theories for explaining how to achieve this largely subjective emotion but, so far, adaptive games are still at its infancy and have typically been considering only one dimension to engage fun: *challenge*. For the existing adaptive games, this usually means that the difficulty of performing game tasks must be in balance with the skills of the player. The goal of such adaptivity is to avoid undesirable 'too easy' or 'too hard' situations.

Such *challenge* purpose has been studied outside the context of adaptive games and within player modeling techniques. Supervised machine learning has been used for this goal. Through analysis of a training data set, consisting of correctly labeled player models, a classifier function is inferred by a learning algorithm (*e.g.* artificial neural networks, decision trees). This classifier function can then be used to model players from real game data sets. Machine learning has been used to model player skills in shooting games [Missura 09], and preferences in strategy games [Spronck 10]. Unsupervised machine learning, where the player models are not labeled *a priori*, has also been proposed in this domain. Player clustering, *i.e.* identifying and aggregating correlated gameplay data, has been applied to classify player styles and preferences [Ramirez-Cano 10]. Furthermore, in terms of fully adaptive games, challenge has mainly steered the methods, algorithms and games analyzed in Sections 2.3 and 2.5.

However, some promising work has already been done around different purposes. Yannakakis and Hallam [Yannakakis 09] propose a methodology for adapting games on the Playware physical interactive platform. The authors explore control of user satisfaction rather than game difficulty, and their testbed is a "bug" (tile) stepping game for children. To model player satisfaction, the authors identify curiosity (the spatial diversity of bugs) and challenge (pace with which bugs appear and disappear) as the main factors. Furthermore, Pedersen *et al.* [Pedersen 10] build quantitative models that predict the player experience in a platform game, to be used in generating levels that are adjusted to these predictions. These models can predict gameplay as being: fun, challenging, boring, frustrating, predictable or anxious. Fig. 2.2 illustrates an example where, after a gameplay session, the system predicts what emotions were experienced by the player.

These approaches show that there is room for going beyond *challenge* as a motivation for steering adaptivity. Magerko [Magerko 08a] argues that players have widely different reasons for playing and that adaptive games should capture and use them, focusing on the players main interests and adapting to match their motivations for playing. Both methods above show promising results in capturing, as Magerko proposed, other dimensions than challenge, as useful indicators of players' motivations for playing. Curiosity, boredom, frustration, predictability or anxiety are powerful features that extend beyond fun or challenge. They can allow for more detailed and flexible mechanisms of steering adaptation and generation.

Affective computing and advances in facial, motion and physiology monitoring



Player Experience

Fun	35.78%
Challenge	37.33%
Frustration	62.91%
Predictability	75.30%
Anxiety	45.93%
Boredom	92.40%

Gap Entropy	0.81
Gap Count	20
Gap Widths	34
Switch	0.51

Figure 2.2: Pedersen *et al.* [Pedersen 10]: predicted player emotions from a gameplay session

can have an important role in steering adaptivity as well. When applied to games, these technologies have the potential to identify the affective states players experience. A better understanding of these can allow for more effectiveness and higher resolution in choosing and designing adaptivity purposes.

Recent research has been done in this direction, through the recognition of steering purposes as challenge [Rani 05], boredom, engagement and anxiety [Chanel 08] and enjoyment preference [Tognetti 10] in adaptive games, using player physiology detection technology (*e.g.* electrocardiograms, galvanic skin response, electroencephalograms, palmar temperature sensors). A more in-depth discussion of the relation between affective computing, physiology detection technology and adaptive games is out of scope here and can be found in [Yannakakis 11].

2.2.2 Serious games and simulations

Serious games and simulations have purposes other than entertainment. For example, they may aim at providing educational or training experiences, where players are required to achieve learning goals in supervised (and sometimes collaborative) environments. In this context, the motivation for steering adaptivity becomes clearer: improve the effectiveness of the knowledge transfer between the game and its players.

Traditionally, to steer adaptivity, research in serious games and simulations has been using a similar approach as in entertainment games: finding a balance between the player's skills and the game challenge level. Reaching this balance remains relevant for serious games and simulations, since it is a straightforward way of simplifying all types of learning goals and styles.

In many serious games, the learning component strongly influences design decisions. For example, the design philosophy of serious games needs to constantly balance *play* (or entertainment) with *meaning* (knowledge transfer) and a strong sense of *reality* [Harteveld 10]. Therefore, in serious games, adapting to specific skills is more important than to the global notion of difficulty or challenge. The learning goals to achieve are usually strongly coupled with the gradual personal improvement of a skill set, most of the times, one skill at a time. In this domain, adaptive games have specialized (and usually *ad-hoc*) approaches, where game components are adjusted to encourage training a specific skill. Adaptivity is steered by a specific skill players need to learn in a particular moment, and influenced by their personal proficiency.

Westra *et al.* [Westra 10], Peirce *et. al.* [Peirce 08], Magerko *et. al.* [Magerko 06] (all further analyzed in Section 2.5), as well as Niehaus and Riedl [Niehaus 09] (discussed in Section 2.3), all propose *personal skill proficiency* as the steering purpose for their adaptivity mechanisms. Another skill-oriented adaptive simulation was proposed by Johnson *et. al.* [Johnson 04], where individual language skills are modeled, determining how a virtual tutor offers guidance to the player. Lane *et al.* [Lane 07] also use a virtual tutor which, constrained by the player's past actions, gives feedback towards a set of skill-based training goals. Martin *et al.* [Martin 10] automatically generate scenarios for serious games using training objectives as the main requirements for generation. Although players are not modeled, these training objectives are also a list of specific tasks (or skills), appropriate for the domain of the game, *e.g.* hit a distant target using an artillery unit.

Some interesting research has been done beyond pure skill modeling and considering other aspects of the learning process. Research on the Crystal Island narrative-centered learning game demonstrates that supervised machine learning can be used to recognize players' affective states [Rowe 09] or model their knowledge [Rowe 10a]. However, future work stills needs to be addressed to apply the recognized data to the adaptation of game content. On a different direction, Magerko *et al.* [Magerko 08b] identify learning styles (*e.g.* explorer, achiever) in users of an educational game; they then adapt the game to better fit players who have those learning styles, to better acquire the desired knowledge. This research shows that steering adaptivity in serious games and simulations can extend further beyond specific skill modeling, to focus on other important features of the player's learning mental process.

2.2.3 Assessment in serious games and simulations

Apart from their purposes, serious games have another differentiating aspect: assessment. Measuring, discussing and reasoning on the gameplay effectiveness is specially important in the simulation domain, since it can lead to reflection and therefore improved learning. However, in this context, assessment has seldom been considered in academic research. In particular, there is no work on combining game adaptivity with assessment. Chen and Michael [Chen 05] have already identified the main challenges that assessment in serious games is facing, namely affecting and improving player experience. The authors suggest that log information and teacher/instructor knowledge should be fully explored and, in some way, incorporated back in the game, to guide its progress.

So far, research in assessment for serious games has been mainly centered on After Action Review (AAR) methods. Still, some results already demonstrate that the direction identified by Chen contains much potential. Lampton *et al.* [Lampton 05] propose an AAR system for military simulations where trainees and trainers assess exercises together. An interesting result was that participants developed innovative ways to use AAR, not only for assessing past behavior, but also for planning new future training exercises. Raybourn [Raybourn 07] proposes a design method for creating training simulations that promote player communication, in-game performance feedback and sharing of strategies. The author focuses on using in-game and AAR assessment information to create an emergent domain culture that could allow the co-creation of future game scenarios.

Some recent research is already incorporating performance logged data to control virtual participants in AAR sessions. Lane *et al.* [Lane 07] proposed a virtual reflective tutor that, given the history of player actions, is able to automatically assess their performance and even conduct an interactive deep reasoning AAR with the player. Core *et al.* [Core 06] and Gomboc *et al.* [Gomboc 05] proposed explainable AI, a game log based system in which AAR participants can directly question virtual characters about their in-game actions, goals and even motivations behind those.

2.2.4 Discussion

With respect to our initial definition of adaptivity's steering purposes, entertainment games and serious games/simulations still form two rather different cases, although both entail valid research challenges that are now discussed.

In entertainment games, some approaches are already being explored beyond the traditional dynamic difficulty adjustment mechanism. A major challenge still lies in exploring even further and materializing Magerko's [Magerko 08a] vision. To adapt better and more, there is a stronger need to capture and be guided by the real reasons why people play. These reasons can be captured by the characteristics and affective states of the gameplay that players expect to experience and be immersed in. For example, a player whose motivations for playing a First Person Shooter (FPS) game are to engage in a specific level of a stressful, scary but low pace experience.

Serious games and simulations are a different case. Due to their specific learning/training goal, many specialized approaches can adapt the game to provide opportunities to develop the most needed skills, at the appropriate proficiency level. However, research shows that there is a need to better account for player individuality. Besides the case of learning styles-based adaptivity [Magerko 08b], Rowe *et al.* [Rowe 10b] also evidence this. The authors investigate individual differences in gameplay and learning during the student's interactions with an educational game. They conclude that learning preferences (student background knowledge and interests) are strongly coupled to the gameplay style (*e.g.* objects used, content read) and need to be considered in game design. The challenge in steering adaptive serious games and simulations still remains in reaching further beyond skill modeling.

By addressing these steering challenges (and related ones) we can reflect on a key research question: *how can we steer the adaptive generation of game worlds*?. Fun, challenge and other gameplay-oriented affective states will typically remain as the *purposes* of adaptivity. However, they need to be better accounted individually, in a more specialized fashion. For example, in Super Mario, an adaptive game could create more hardly accessible coins for a player who is excellent at collecting them.

The surveyed research shows that modeling player skills, preferences, styles and learning goals are effective methods to support such adaptation purposes. We are confident that the same would hold for more specialized purposes. The specificity of the adaptation could come from the steered adaptation and generation engine, like in the example above. Another major challenge in this direction lies in supporting these mechanisms in a game domain independent fashion, so they can be re-used and consolidated.

In addition to player skills, preferences, styles and learning goals, assessment of past performances can also play a role in adaptive simulations. In this domain, there is typically plenty of valuable information emerging from game logs and AAR sessions. Using this information as a source to steer adaptivity seems a promising, unexplored area. Interesting research opportunities exist in using assessment information to, for example, re-generate 'try again' game missions, adapted and focused on what the players failed during the previous session. So, offering an adapted re-generated 'game session' could simultaneously allow a better understanding of what went wrong, and better opportunities to succeed.

2.3 Adaptive game components

After discussing the *purposes* of adaptivity in the previous section, we now turn our attention to one of our key research questions: adaptivity's *features*. Potentially, all components that are considered at game development can become adaptive. In fact, dynamically adjusting (i) game worlds and its objects, (ii) gameplay mechanics, (iii) non playing characters (NPC) and AI, (iv) game narratives, and (v) game scenarios and quests, all can contribute to offer an individualized gameplay experience. Table 2.1 illustrates how surveyed work is distributed according to game components and domain.

Gameplay mechanics, *i.e.* how game elements can work, including actions like running or shooting [Brathwaite 09], have already been made adaptive, in commercial games. In *Max Payne* [Remedy Entertainment 01] (illustrated in Fig. 2.3), a mechanism unknown to players altered the level of mechanics like player aim assistance,

Table 2.1: Classification of surveyed work according to adaptive components and Industry / Academia domains

	Commercial games	**Academic research**
Game worlds	[Valve Corporation 09]	[Nitsche 06], [Togelius 07], [Shaker 10], [Jennings-Teats 10], [Kazmi 10], [Shaker 12]
Mechanics	[Remedy Entertainment 01]	[Hunicke 05], [Yannakakis 09], [Magerko 08b], [Kazmi 10]
AI / NPC	[Nintendo EAD 08, Konami 07]	[Westra 09] , [Peirce 08], [Bakkes 09b], [Bakkes 09a], [Hartley 09], [Spronck 06], [Kazmi 10], [Andrade 06], [Olesen 08]
Narratives	[Valve Corporation 08, Valve Corporation 09, Quantic Dream 10]	[Thue 07], [Barber 07], [Mott 06], [Sharma 07], [Fairclough 06]
Scenarios/quests		[Magerko 06, Niehaus 09, Sullivan 10, Pita 07, Ashmore 07]

according to individual skills (thus adjusting shooting difficulty).

Traditionally, adaptivity has been mostly researched and applied within the *AI*

Figure 2.3: Scene from *Max Payne*, by Remedy Entertainment [Remedy Entertainment 01]

domain, specifically towards NPCs, since behavioral adaptation is a strong means of displaying intelligent behavior. In *Mario Kart Wii* [Nintendo EAD 08], rubber band AI techniques are used to increase the opponent NPC abilities when the player performs too good. *Pro Evolution Soccer 08* [Konami 07] introduced Teamvision, an adaptive AI opponent system that changes its tactics and strategy to suit the player style and explore his weaknesses. In academia, and as identified in Table 2.1, several techniques have been proposed to support adaptive AI that recognizes the player actions and responds by adjusting NPC behavior. Also, academic research on AI adaptation focuses strongly on the pedagogical serious games domain, due to the extensive use of NPCs in learning contexts (*e.g.* tutors). Several of the techniques surveyed in Section 2.5 are applied to AI adaptation, both in entertainment and serious games.

Adaptivity has also been applied to *game narratives*, both in the commercial and academic domains. Games can become more personal when the progressing narrative builds up in a unique fashion, fitting the players' behavior. Valve's *Left 4 Dead* series [Valve Corporation 08, Valve Corporation 09] introduced procedural narrative as a technique to generate sequences of events, adapted to the pace and behavior of the player. An AI Director analyzes players behavior (*e.g.* if they were particularly challenged by one kind of enemy) and adds subsequent events (*e.g.* spawning that enemy). According to Valve [Newell 08], this mechanism serves as a story-telling device (at least, in simple narrative domains as most FPS games are) because players can experience some notion of intentionality on the opponents' side. *Heavy Rain* [Quantic Dream 10] is an interactive drama game that focuses on personal gameplay, where all the specific decisions each player takes are analyzed, in a more complex way than before, to determine the narrative and outcome of the game.

In academia, there is a strong interest in interactive narratives, story-based experiences which typically use game technology, both for entertainment or pedagogical purposes. Roberts and Isbell [Roberts 08] have recently surveyed interactive narratives and drama management systems, identifying, among other aspects, their adaptive capabilities. Here, we present only a brief overview of these systems. For a more detailed discussion, *e.g.* on concerns as the use of centralized manager agents vs. multi-agent networks, Roberts and Isbell's survey is recommended.

Barber and Kudenko [Thue 07] researched the generation of dilemma-based interactive narratives. A model of player behavior under specific dilemmas is used to estimate and select difficult dilemmas, which a planner weaves together to form a story. Mott and Lester [Mott 06] use a dynamic decision network as a planner for creating interactive narratives. The decision network contains nodes for the player's goals, experiences and relationships, thus influencing decision making. In Sharma *et al.*'s drama management system [Sharma 07], player preferences are determined by an explicit case-based player model, derived from the behavior of earlier players. This model guides generation towards stories that fit those preferences. Fairclough [Fairclough 06] also uses a case-based approach, but to synthesize stories from a knowledge base, constrained by the player's evolving relationship with NPCs. Finally, Thue *et al.* [Thue 07] present an interactive narrative generation system which models the player's style according to five predetermined player types. Events are annotated with their appeal for each player type and are selected accordingly for inclusion in the narrative.

Game scenarios and quests only recently started to become a target of adaptivity research. Game scenarios and quests both describe the flow of events and actions within a game but they are primarily used, respectively, in simulations and entertainment games. Generation of personalized game quests is already being researched and is discussed in detail in Section 2.4.2. As for game scenarios, they highlight the importance of adaptivity in the simulation domain. Niehaus and Riedl have recently proposed a methodology for adapting game scenarios to suit players learning goals [Niehaus 09]. A Scenario Adaptor adds, removes or replaces abstract game events, guided by a mapping between a world domain-knowledge base (*i.e.* the dynamics of the simulated world events) and a lifelong-learner model, which tracks a learner and chooses the next training objectives that will help him advance. Earlier research from Magerko *et al.* [Magerko 06] also adapts game scenarios, and it will be discussed in detail in Section 2.5.

As for results on adaptive *game worlds*, they are very scarce. The only example we found of game world adaptivity is in the commercial game *Left 4 Dead 2* [Valve Corporation 09]. According to the developers, the layout of certain sections of levels is dependent on the player's performance [Walker, J. 09] (a graveyard with a simpler layout for underachieving players is presented as an example). Being a recent commercial release, the game's publishers have not yet disclosed any technical details nor the reach of this adaptivity mechanism, in terms of accounting for how much of the content is static or dynamic. It is therefore still unknown which player modeling or procedural content generation techniques are used, if at all. As for academic research, several projects, mainly aimed at player modeling and difficulty adjustment techniques, are focusing on adapting 2D game level structures, as discussed in Section 2.5. This simple game levels are still far from being compared to the complexity and richness of modern game worlds. Although not fully adaptive, Charbitat [Nitsche 06], further analyzed in Section 2.4, is the only example we are aware of where procedural generation of complex game worlds is somehow influenced by player performance.

2.3.1 Discussion

With this survey, we identified the adaptivity *features* still lacking on broad, consolidated and integrated research focus: modern *game worlds* and *scenarios* (or quests). Research in game scenarios has already shown promising results. Niehaus and Riedl [Niehaus 09], and Magerko *et al.* [Magerko 06] are good examples of the advances achieved so far. However, there are still some open challenges: (i) reach beyond skill-driven adaptivity (as discussed in Section 2.2), and (ii) integrate scenario with world adaptation/generation.

As for game worlds, and beyond the valuable research performed on classic simple level structures, results are scarce. Considering more complex and modern game worlds would offer more opportunities to achieve the open challenges in steering adaptivity (Section 2.2.4). More complex and immersive game worlds are not only richer to players, but also offer more opportunities to adapt, *i.e.* more (combinations of) content to generate. Adaptive object placement in a 3D environment or the generation of 3D game spaces/maps are examples of possible open challenges.

The importance of these two components, particularly if they are integrated, can be highlighted through their definitions. Game worlds are the virtual environments within which gameplay occurs, with their geometry, geography, layout and objects. Game scenarios are the framework for the global progression within a game level, with their initial settings and the logical flow of events and actions that follow [van Est 11]. As such, the fulfillment of a game scenario within a game world defines and characterizes most of the player experience. Integrated world and scenario adaptivity seems therefore very likely to solve the shortcomings identified in Section 2.1, certainly offering meaningful possibilities for affecting player experience.

Currently, game worlds are created during the design stage, prior to game release. In that process, games and simulations occasionally use procedural generation algorithms to automatically create some of the game world elements, with techniques widely researched in academia, like the ones surveyed in Section 2.4. As for game scenarios, they are typically created during the gameplay programming stage, when scripts and code define the flow of events for the game. A major challenge in automatically authoring game worlds and scenarios, as in fact with all game content, lies in delaying its generation until the game is running. This challenge is essential for adaptivity, since the creation of content that is adjusted to players relies on analyzing their in-game performance. There are two main *methods* to tackle the challenge of supporting adaptive game worlds and scenarios, through delayed authoring: (i) off-line (pregb-game) customized generation, and (ii) on-line (*i.e.* in-game) adaptivity. In the next two sections we will survey the present state of research on each of these topics, and how they confirm that adaptive game worlds and game scenarios raise very promising and challenging research questions.

2.4 Off-line adaptivity: customized content generation

Off-line adaptivity implies that adjustments are made considering player-dependent data, but prior to initiating any gameplay. The typical example of its application would be the processing of player data and game adjustments during the loading stage of a game level. Therefore, off-line adaptivity involves mainly a generation challenge.

Automatic content generation can therefore play a significant role in off-line adaptivity. Research results in this field are particularly promising towards *customized content generation*, a *method* for the automatic creation of virtual game worlds, adjusted to better suit players. We believe the same principles can be extended and applied to what occurs within these worlds, *i.e.* to game scenarios, even though their off-line generation has been less investigated than that of game worlds.

2.4.1 Game worlds

Previous work in automatic content generation has traditionally relied on procedural methods and has often succeeded in creating visually convincing game environments. For the public eye, procedural generation has been successfully associated with games, due to *Elite* [Braben, D. and Bell, I. 84] or, more recently, *Spore* [Maxis 08]. The latter extensively uses procedural generation for player-designed creatures, animations and planet textures.

Regarding game worlds, many different procedures have been proposed to automatically create content such as terrain, trees, plants and urban environments. Procedural methods were recently surveyed and discussed by Smelik *et al.* [Smelik 14], who conclude that a common shortcoming in traditional methods is the lack of control over the generated output. Therefore, researchers are now aiming at more controllable procedural methods, seeking to allow designers to intuitively steer content generation.

In this direction, interesting work has been done in the generation of 2D platform game levels. Compton and Mateas [Compton 06] use context-free grammars to generate platform levels, organized in patterns and branch structures. The generated level is controlled by a hill-climbing algorithm that adjusts patterns to suit a target controllable difficulty. Smith *et al.* [Smith 09] further developed these concepts, allowing designers to directly constrain properties in the generated platform levels (*e.g.* level path, jump rhythm and frequency, etc). Sorenson and Pasquier [Sorenson 10] propose another approach where genetic algorithms are used to evolve 2D game levels towards satisfying designer constraints. An interesting result lies in how they evaluate generated levels: they are subjected to a fitness function that rewards levels based on how fun (in this case, challenging) they are. These results show that generation of 2D level structures has succeeded in considering important adaptivity concepts such as difficulty, challenge or fun.

The generation of modern 3D game worlds is facing other issues, more related with intuitive and interactive control. In this domain, Müller *et al.* [Müller 06] proposed the use of shape grammars to generate highly detailed cities. The grammar uses context sensitive rules to iteratively evolve building design, by creating more and more detail. Users can control the generation of a city using their *CityEngine* system, allowing them to create and edit grammar rules, in a similar way to using scripting languages. Fig. 2.4a shows a model for the ancient city Pompeii, as generated by *CityEngine*.

Recent research has focused on creating new methods for designers to control game world generation, more intuitively than shape grammars. Doran and Parberry [Doran 10] propose an approach where terrain elevation heightmaps are generated by

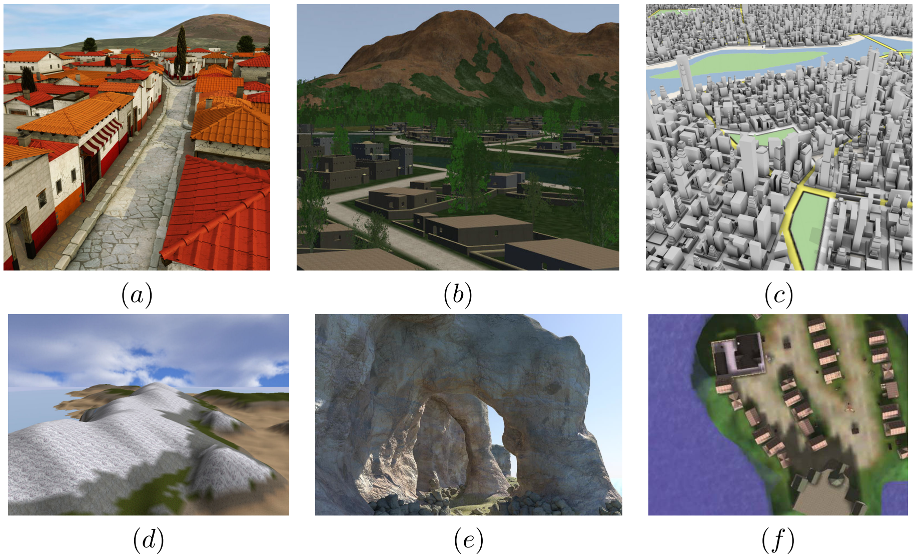


Figure 2.4: a) urban environment generated with *CityEngine* [Müller 06]. b) virtual world generated with *SketchaWorld* [Smelik 11b]. c) road network and corresponding 3D city geometry, generated with [Chen 08]. d) height-map, generated with [Doran 10]. e) complex terrain, with arches, created with [Peytavie 09]. f) top view of town, generated with [Bielikova 08].

independent software agents, with different roles for coastlines, beaches, mountains, hills and rivers. Designers are responsible for defining terrain features that constrain the amount of agents, their lifetime and actions and, thus, the way the terrain is generated. Peytavie *et al.* [Peytavie 09] present a framework for generating complex terrains that include overhangs, arches, caves and different materials such as sand and rocks. Designers can control the terrain generation by sculpting bedrocks, modeling cracks, fractures and tunnels, adding granular material and controlling erosion tools. Chen *et al.* [Chen 08] use tensor fields to guide the generation of street networks. Users can control the generated street network by placing basis tensor fields, using tensor field patterns, smoothing fields to reduce its complexity, brushing the field to orient streets or applying noise to make the road network less regular. Fig. 2.4c, 2.4d and 2.4e show, respectively, a road network created by Chen *et al.*, a height-map generated by Doran and Parberry, and a complex terrain modeled by Peytavie *et al.*.

Even more interactive and user-centric methods have been proposed to control automatic content generation by: sketching the silhouette and bounds of a mountain in a 3D interface [Gain 09], brushing and sculpting outdoor terrains [de Carpentier 09] and sketching roads, which are automatically generated to fit with the surrounding environment [McCrae 09].

Some recent research results have already shown that control over the generation process can extend beyond this type of interactive modeling of geometric world features. Bielikova *et al.* [Bielikova 08] propose a system for generating educational game content: quests, NPC, virtual worlds (see example in Fig. 2.4f) and narratives. In this case, domain experts, *i.e.* teachers, and not designers, control content generation. Teachers can select pre-created game objects, add new learning content to them and create relationships between objects. Knowledge about objects and their relationships is the basis for solving and generating all the appropriate content. These results offer another valuable contribution: control on the generated content is applied at a higher level than geometric features, by using knowledge on objects and their relationships.

Nitsche *et al.* [Nitsche 06] introduce a case study for the procedural generation of game worlds based on the gaming style of its players. In *Charbitat*, players steer the generation of an infinite world through their in-game actions. The game world is split into individual tiles and each new tile is generated using noise functions and filters, where the underlying seed value is calculated based on player-dependent character data, *i.e.* his actions. Players are involved and conscious of this process: they can voluntarily influence the world generation in different directions as they please. Although this is an on-line method, this guided generation nature relates better with the methods and requirements for off-line adaptivity.

Both [Bielikova 08] and [Nitsche 06] show that automatic generation of game worlds can be controlled on a higher level (when compared to geometric features), and can be made dependent on player data. Both results seem successful advances towards customized content generation.

2.4.2 Game scenarios and quests

Off-line automatic generation of game scenarios and quests has not been a subject of much research, especially when compared with on-line scenario adaptivity (Section 2.5). The term game scenario, *i.e the global progression within a game level, including its initial settings and the logical flow of events and actions that follow*, is mainly used in serious games and simulations. Its entertainment game equivalent, game missions or quests, also structures a sequence of events and actions, normally associated to a game task that must be completed.

Research in this field shows that there is a growing interest in creating player-centric quests that provide personalized gameplay. Sullivan *et al.*'s Grail framework [Sullivan 10] is aimed at providing customized quests, through on-line player-centered adjustments (better analyzed in Section 2.5), but it also includes an authoring tool for designers to control quest generation.

Although the following two methods are in essence also on-line based, their simple definition of quest avoids the usual design requirements of on-line methods (*e.g.* performance or consistency concerns). Therefore, they relate closely to possible off-line techniques. Pita *et al.* [Pita 07] propose a system to dynamically generate quests in persistent Massively Multiplayer Online Role-Playing Games (MMORPG). Quest generation creates valid game goals, which are unique for each player and game. Quest uniqueness is ensured by three player-centric features that constrain the generation process to produce relevant quest paths: the memories (past quests) of the player, his relationships to the character assigning the quest, and player attributes (needed to complete quests). Ashmore and Nitsche [Ashmore 07] also investigate player-centric quest generation. They propose a new quest generator to include in the previously discussed *Charbitat* [Nitsche 06] system. Quests consist of key and lock puzzles (a key must be found to unlock an obstacle) and the generation process places within the game world, both the locked obstacle, its key, and the challenges along the way to obtain it. Quest generation occurs during the generation of a new world tile: possible locations for keys and locks are scored by evaluators that are highly dependent on the procedurally generated tile. In the Charbitat case, quests become unique for each player because they are influenced by the game world which was itself generated in such a customized fashion.

These results evidence some of the potential in integrating and influencing game quest generation with the surrounding game world. As stated in Section 2.3, integration with the game space is an important aspect to be considered in quest generation. In Pita's case, quests are generated in a game world that was manually designed, before-hand. In Ashmore's approach, the game world is first procedurally generated and is then evaluated for placement of quest elements. Though not adaptive in any way, Dormans work [Dormans 10] is a good example of a totally different approach, a constructive integrated one. The generation of 2D action-adventure game levels is broken down into two steps: a graph grammar generates mission structures that are used in an extended shape grammar, which grows a space that accommodates the generated game mission.

Off-line generation of game scenarios, as defined earlier, is still far behind these concepts of customized quests or missions. Research in off-line scenario generation is still more focused on the methods, *i.e.* on *how*, to generate and less on its purposes, *i.e.* on *what for*, *e.g.* steering them to be player-centric. As mentioned in Section 2.2, Martin *et al.* [Martin 10] generate game scenarios for serious games. They use functional L-systems, a variant of formal grammars, to write generation rules which can expand training objectives into generated scenario elements, *i.e.* the initial settings and the progression of game events. Hullet and Mateas [Hullett 09] also generate game scenarios from pedagogical goals, but using a planning system that decomposes pedagogical goals into tasks, subtasks and methods, which encode knowledge to achieve that goal state.

Both approaches generate game scenarios from goals that capture which skills the players should apply during the game. However, in both methods, these declarable goals are simply a low-level and domain-dependent set of features that are implied by the higher level desired gameplay skills. For example, Hullet explicitly declares the goal *'a room should be blocked'* to implicitly capture the skill of breaching walls to rescue victims.

2.4.3 Discussion

In this section we surveyed research related to customized content generation, namely off-line procedural generation of 2D game levels, 3D game worlds, quests and game scenarios.

The surveyed methods show that the generation of 2D levels is already capable of being controlled, or at least evaluated, by the same kind of criteria currently used to steer adaptivity: difficulty, challenge, fun. These results highly encourage the further use of player data, *e.g.* their preferences or performance, for controlling the procedural generation of game levels. Even though level generation for the platform game genre is less complex than the generation of modern game worlds, the same conclusions could still hold for the latter.

Research shows that this is still far away, since the generation of complex game worlds is facing other issues. The main challenge is to enable designers to control the generation process. Controllable content generation is enabling procedural methods to become more flexible and accurate. While maintaining its automatic nature, these methods are allowing game designers to steer automatic content generation by means of a better expression of their intent. Although these results are aimed at the design stage, they seem promising steps towards customized content generation, as they allow procedural methods to be interacted with and controlled.

Control of content generation at design time is also the key for an unexplored research direction in this field: authoring adaptive generation in games. All of the surveyed methods in this Chapter rely on a technical approach, where adaptivity is *programmed* into the game code and not *designed*. An open challenge is to support game designers to author adaptive generation, *i.e.* enabling them to link what and how game content should be generated to individual player requirements. Such link could be created by controlling content generation from a higher level (when compared to geometric features) and making it dependent on player data capturing adaptivity's purposes (Section 2.2.4). A non-technical type of control, from a higher level of abstraction, is essential to allow interactive design. Interactively creating such links would enable game designers to specify and author adaptivity, one of our key research questions (Chapter 1). The research of Bielikova *et al.* and Nitsche *et al.* showed encouraging results in controlling the automatic generation of game worlds from a higher level.

Regarding quest and scenario generation, results showed that customized generation is becoming more relevant, and it can be a successful mechanism to engage players in more enhanced, interactive and personal experiences. However, the methods surveyed are still somehow rudimentary, due to their *ad-hoc* nature. For example, quests are defined in an elementary manner, and generation is constrained to only one aspect of what a quest can include: goals to accomplish and locations for objects, in the cases analyzed. Furthermore, the analyzed methods show that the challenges ahead are the same as with game worlds: (i) considering higher level skills or goals (or the learning preferences discussed in Section 2.2) in an explicit way and (ii) taking advantage of player-dependent data. As stated in Section 2.3, we think that fully integrating world and quest/scenario generation is a potentially important milestone.

Current research is already tackling some of the challenges identified above, and its methods could be valuable to future work in customized content creation. Semantic and declarative modeling techniques are already capable of controlling procedural methods by embedding and interpreting higher level knowledge in virtual objects. Tutenel *et al.* [Tutenel 08] define object semantics as all information, beyond its 3D model, related to a particular object within the game world (*e.g.* functional information like how to interact with it, possible relationships with other objects, etc). With semantic modeling, object relationships, features and other semantic information can be used to guide the layout generation of a game world, whether designing it manually or creating it procedurally.

Bidarra *et al.* [Bidarra 10] introduce declarative modeling of virtual worlds, explaining how semantics can help designers to create virtual worlds by declaring *what* they want to create, instead of *how* to model it. Such declarative modeling enables designers to control and constrain virtual worlds, through semantic specifications that describe what the virtual world and its objects should be. Fig. 2.5 illustrates how this semantic level, presented to designers, is used to control the procedural level. This scheme differs from conventional procedurally-based modeling, sporadically used by designers and technical artists, in that it incorporates a semantics layer, between the designer and the procedural techniques. This semantic level provides designers with a powerful front-end that steers the underlying procedural level, while encapsulating the complexity of the latter.

Many of these methods have been integrated in *SketchaWorld* [Smelik 11b], a prototype system for declarative modeling of virtual worlds. In this declarative approach, designers state their intent by specifying the high-level features a virtual world should have, *e.g.* the layout of the landscape or the population size of a city. Designer's intent is used to generate a matching 3D virtual world, where each specification is procedurally expanded to a visually convincing terrain feature. Within this declarative approach, interactions between terrain features are automatically solved using virtual world consistency maintenance, which consists of a combination of semantic definitions of the geometric and functional relationships between terrain features, and a set of generic resolution rules. A virtual world created with *SketchaWorld* is shown in Fig. 2.4b.

Semantic and declarative modeling can already help in tackling some of the challenges we identified throughout this section. The previously explained semantic layer deals with all high-level information relating to virtual world objects at the semantics level. This information helps conveying the meaning and the role of an object in the virtual world, and consists of generic descriptions of classes of features, including attributes, properties, roles, relations, etc. This encourages the incorporation of further semantic information about player dependent gameplay purposes, and how these can be used to control object generation. For example, if a player needs

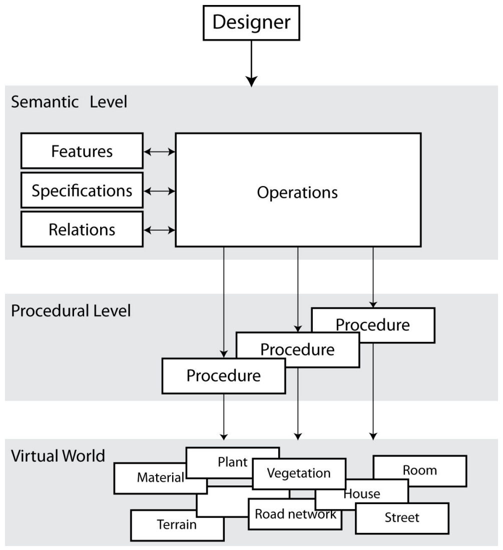


Figure 2.5: Generic approach of declarative modeling of virtual worlds [Bidarra 10]

obstacles in a race track, the semantic layer could indicate *which* and *how* obstacles can be used with that player. Furthermore, this type of semantic information could potentially be considered as the basis for authoring adaptive generation, by supplying a vocabulary to declare dependencies between player data and content, as discussed earlier.

Finally, declarative modeling already includes semantics-based mechanisms to check and solve conflicts in procedurally generated worlds. This shows that these techniques can be flexible enough to handle conflicting contexts, like those which would likely arise when integrating player-centric with designer-centric purposes, and virtual worlds with game scenarios.

The current state of research in semantic and declarative modeling, however, does not answer all of the issues discussed in this section and many challenges still remain open. Among them, supporting the generation of game scenarios in a similar fashion (enriching them with an analogous semantic scheme), integrating these with virtual world generation, and measuring player data into valid semantic knowledge, these are some of the issues that need to be addressed to consider semantics as a relevant technique to customized content generation.

2.5 On-line adaptivity

As mentioned in Section 2.3, off-line customized generation is not the only *method* to support adaptivity through delayed authoring. On-line, or in-game, adaptivity is the term many researchers use to describe the ability of a game to adjust to its players, in real time, as they play. Although this kind of adaptivity is still a recent research area, there are some significant results worth discussing here.

Most adaptivity research focuses on a low level, *i.e.* adapting specific game elements through non-integrated approaches. The traditional approach to adaptive games has been the dynamic difficulty adjustment (DDA) mechanism. Non-PCG DDA approaches have been used to influence players health or ammo in a first-person shooter, depending on their individual performance, through probabilistic models [Hunicke 05], to evolve opponent AIs in real-time strategy games, through neural-evolution methods [Olesen 08] and to adapt intelligent agents behavior to fit the players skills, using reinforcement learning [Andrade 06].

However, Charles *et al.* [Charles 05] proposed a high-level framework to explain how on-line adaptivity should be supported in every domain, in an integrated manner. This framework captures the main abstract ideas and approaches that are currently adopted throughout this research area. A model of the player is used to capture the player habits and skills, and the player performance is monitored and compared with the model, while playing. Whenever an adaptation of the game is identified and performed, the framework measures its effectiveness, which can lead to either a new adaptation or an update of the player model.

Currently, on-line adaptivity mostly acts as a sandbox for researching new artificial intelligence concepts and methods. As such, most work in this field focuses on adjusting NPCs or other intelligent game agents to better suit players or even offer a more challenging game.

Peirce et al. [Peirce 08] propose the ALIGN system as an approach for non-invasively adapting NPCs behavior to enable a personalized learning experience. ALIGN's architecture separates the logic of generic adaptation from game specificities, so that game logic and adaptation are independently authored and operated. Adaptive Elements (AEs) are the basic components that support possible adaptations. AEs are pre-created and annotated with metadata describing both the game settings in which they can be used, and the abstract outcome of their use. Separate inference engines translate game events to AEs and create specific in-game interventions to match the selected AE. Each in-game intervention is influenced by a set of rules that examine a player model and determine the desired adaptation outcome.

Westra *et al.* [Westra 09] use agent organizations to adapt (in-game) the behavior of game elements in serious games. Uncentralized and independent (learning) agents choose the tasks to be performed by individual game elements, *e.g.* a burning fire or NPCs. Possible behavior variations for all agents are implemented *a priori*, using domain experts knowledge. During the game, each agent infers and proposes possible actions, based on its own in-game goals. The agent organization framework mediates this autonomy, by controlling which behavioral adaptation occurs for each agent. The agent organization framework coordinates individual adaptations into a combined one that adjusts the global behavior of game elements to fit the player skill level and a coherent storyline. For this coordination, Westra uses a player model that estimates the skill levels of each player. This model is continuously updated to accurately steer the agent organization framework.

Bakkes *et al.* [Bakkes 09b] focus on adapting an AI-controlled opponent in a particular real-time strategy game. In this case, on-line adaptivity takes place at the opponent AI, so that it can learn from its mistakes and act more effectively. The authors propose an approach where domain knowledge is gathered automatically by the game AI to form a case base (*i.e.* a compilation of solutions of similar past problems), which is exploited immediately to evoke effective behavior. The case base is compared with observations from previous games to allow improvement on past behaviors. Preliminary research [Bakkes 09a] has been done to incorporate opponent modeling, in this case used for the AI to gain a competitive advantage. Opponent models are established automatically, through clustering of strategic feature data in game observations. Past game observations are classified with such models, allowing a better matching with the case base.

On a similar direction, Hartley and Mehdi [Hartley 09] also use a case-based approach that allows NPCs to learn, while playing, from the player actions, adapting the challenge level in the game. Game observations are gathered in cases that take the form of player state and action pairs. Matching these observations with previously registered cases, can be used to predict the next state-action pair and, therefore, enhance the NPC decision making. Results show that this approach succeeds in predicting player movement and actions in a FPS game. Therefore, despite its adaptivity focus on NPC (instead of player) goals, this case matching algorithm provides a method for adjusting the game according to player actions.

Dynamic scripting [Spronck 06] is another technique proposed for adapting game AI, adopted for dynamic difficulty adjustment to the player skills. This learning technique is able to generate scripts (sets of behavioral rules), from rulebases associated with NPC classes, in order to control NPC behavior. Each rulebase comprises a set of manually designed rules and the probability that a rule is selected for a script is influenced by an attached weight value. Weights are updated according to their success rate in the game, which includes maintaining an even and challenging game for players.

Some promising work has already been done in directions other than AI and NPCs. Adaptive (simple) game worlds and levels are starting to be researched. Togelius *et al.* [Togelius 07] propose an approach for generating, on-line, tracks for a racing game. Their goal is to augment player satisfaction, by creating a track that evolves with the player's characteristics. A player model is implemented by a neural network-based controller which infers and simulates the behavior of the real player. This player model is used to predict entertainment levels of specific players and decide how to evolve the track. Tracks are initialized as b-splines with 30 segments and they evolve through a mutation done by perturbing the positions of their control points. Although the focus is on 2D racing tracks, this work shows promising results regarding the use of player modeling to generate adapted game content. Search-based procedural content generation is also used similarly for creating personalized *Super Mario Bros* platform levels [Shaker 10], with the use of exhaustive search algorithms.

Jennings-Teats *et al.* [Jennings-Teats 10] also focus on dynamically constructing 2D game environments that are adapted to players. In this case, the *Polymorph* algorithm generates 2D platform levels, as you play, driven by dynamic difficulty adjustment to the player skills. A statistical model of difficulty and a model of the player's current skill level are used, through mass data collection and machine learning techniques, to select the appropriate level segments to generate for each player.

In the same direction, Shaker *et al.* [Shaker 12] have proposed to generate platform levels for *Super Mario Bros*, using grammatical evolution. This technique results from the combination of an evolutionary algorithm with a grammatical representation for automatic design. In this case, design grammars represent the underlying structure of game levels. The grammatical evolutionary algorithm uses models of collected player experiences as the fitness functions to search through the generative space of the design grammar. The authors go beyond DDA, optimizing (off-line) the game levels to improve engagement, frustration and challenge.

Kazmi and Palmer [Kazmi 10] also direct their research to adapting game environments. They present a case study for a prototype of an adaptive FPS gaming environment. Player actions are recognized through a finite state machine approach, by which discrete actions reveal the skill level of players. Adaptation mechanisms try to make the game harder for players identified as experts and easier for beginners. In this research, finite state machines have also been used to implement all possible adaptations, *i.e.* adjustments on NPC behavior and movement, weapon mechanics and game level geometry. Although this approach was mainly centered on adapting NPC behavior, the authors successfully explore other alternative ideas. They implemented a simple 'Modify Geometry' mechanism that dynamically changes the game environment so that it becomes more difficult to navigate safely. They conclude that the 'Modify Geometry' mechanism provides the most significant impact on player satisfaction.

In a different direction, some research has been done towards on-line adaptivity in quests and game scenarios. Magerko *et al.* [Magerko 06] ISAT project uses an intelligent director agent for customizing simulation training scenarios to suit individual trainees. A skill model captures player proficiency levels in domain-specific skills, by monitoring and rating the trainee's actions. Scenarios are identified as sequences of plot points, *i.e.* actions, events and skills involved in them, which are selected, at run-time, for inclusion in the simulation. The director selects plot points by matching the list of tested skills with the current state of the skill model.

Sullivan *et al.* [Sullivan 10] also proposed a centralized approach, the Grail Game Manager, a run-time manager which dynamically generates quest structures using the player's history and current world state. This rule-based system is able to decompose quests (from a quest library) into separate entities (goals, actions, rewards, NPCs, dialog options) that can be dynamically recombined upon generation. This process filters possible quest entities through pre-conditions based on player history and current world state, thus creating a personalized experience.

2.5.1 Discussion

As surveyed above, current research results show that on-line adaptivity is mainly concerned with adjusting challenge levels of NPCs and game AI. In these approaches, on-line adaptivity is still characterized by a certain degree of predictability, since some of the analyzed methods require all possible variations to be created *a priori*. Performance and control over the game are the reasons why this balance, between static and dynamic techniques, is a recurring and important challenge in on-line adaptivity.

The current scope (AI and NPCs), purpose (challenge level balance) and techniques (combination of predefined content) in these approaches show that on-line adaptivity is in its first steps. Integrated approaches, embracing on-line adaptive game worlds and scenarios, are still far away. Confirmation of this is the fact that player modeling and monitoring is still considered on an individual case, without sound and common theoretical foundations.

However, recent work has broaden the focus to adapting game environments and other game elements. Although, for example, Kazmi's 'Modify Geometry' mechanism was simple and applied to only one type of situation, one can easily foresee the potential of adapting more than just the behavior of intelligent agents. Procedural content generation (as surveyed in section 2.4) is becoming more powerful and can have a role to play in on-line adaptivity as well. An example of such potential is evidenced by Kenneth Stanley's Galactic Arms Race [Hastings 09], a multi-player game where players control a spaceship and collect weapons throughout a game world. Weapons are procedurally generated, at run-time, based on which weapons have been selected and used before by players.

Results both in 2D game worlds and quest/scenarios confirm this observation: procedural content generation is becoming more and more on-line efficient and player-centric. This demonstrates that online adaptivity shares the research opportunities of customized content generation (Section 2.4.3). Specifically, on-line generative methods still also need to be steered/controlled by (player-dependent) data which can be related with adaptivity purposes (Section 2.2.4). Furthermore, creating a non-technical type of control, using parameters on a higher level of abstraction and intrinsically related with gameplay, will enable that such type of adaptivity can be authored by game designers.

Finally, modifying landscapes and topography of virtual worlds has already been suggested as a valid direction for on-line adaptivity [Charles 05]. However, future methods for dynamically changing game environments, on a world scale, must tackle important challenges to be successful. Among them, maintaining coherence (*e.g.* mountains cannot magically change shape) and ensuring performance and scalability are important aspects, so that this type of adaptivity does not undermine the player experience.

2.6 Conclusions

In this chapter, we surveyed the present state of adaptivity in games and simulations. We focused on the *purposes*, *features* and *methods* that have been proposed so far to support adaptive game technology, from both academia and industry. Our goal was to investigate and conclude on the research opportunities to improve the development of adaptive games (Chapter 1).

Our first conclusion is that, regarding its *purposes*, adaptivity is already establishing itself as a rapidly maturing field. Current advances, both in industry and academia, indicate good results in not only adapting towards an optimal challenge level, but also towards other affective states like fun, frustration, predictability, anxiety or boredom. With simulations, research is already successful in adapting to fit specific skill levels and incorporate learning styles.

Concerning the *features* of adaptivity, we have concluded that a large community both in industry and academia has already been focusing on game mechanics, AI, NPC and narratives. Fewer research groups are already focusing, with success, on adaptive game scenarios or quests. On the other side, concerning adaptivity in modern and complex game worlds, many research questions are still unanswered.

Furthermore, there is a lot of potential not only in adaptive game worlds, but particularly in their integration with adaptive scenarios/quests. Acting upon these, in an integrated manner, can create plenty of (yet unexplored) possibilities for improving gameplay.

Regarding the *methods* which can support adaptivity, some important advances have already been achieved with off-line and on-line techniques. One of the most promising *methods* is the procedural generation of off-line (*i.e.* pre-game) content that is customized to fit each player. Procedural content generation is becoming more controllable, although mainly through control over geometric features of that content. Some preliminary work has been done to: (i) incorporate player data, (ii) control generation using high-level information (*e.g.* object metadata, semantics) and (iii) integrate game level and event generation. Further advances are being achieved in generating personalized basic quests, for entertainment games, and generating scenarios from declarable learning goals, for serious games.

On-line (*i.e.* in-game) adaptivity is also an essential *method* to consider. Current research is succeeding in using player models to control the adaptation of NPC run-time behavior. Moreover, promising work is being done on dynamically constructing game scenarios, and even of game environments, that are adapted to the player in-game performance. For adaptivity purposes, on-line adjustment of game worlds and scenarios is likely to achieve better results than with NPCs. However, new methods will need a paradigm shift, already encouraged by the recent advances in procedural content generation: from searching and selecting among predefined solutions to generating dynamic emerging ones.

Fig. 2.6 summarizes the challenges of this field and how they relate to each other. For both entertainment games and simulations, the main overall challenges include capturing and incorporating higher-resolution player data, including gameplay expectations, broad learning preferences and assessment data, to generate (off-line or on-line) more complex game worlds and scenarios. An underlying challenge lies in enabling designers to author such systems in an interactive fashion.

These conclusions allowed us to focus our research contributions in the adaptive generation of game worlds. In this thesis, we are interested in improving upon the present off-line and on-line adaptation methods. From this survey, we concluded and formulated our key research questions: how can the adaptive generation of game worlds be controlled/steered? Which complex and modern game world features can be generated? How can this process be authored by game designers? And how can it be generic enough to apply to different games?

We concluded that recent research results in semantic modeling can offer a promising starting point to answer these questions. Object semantics have the potential to be made dependent on player data and control content generation in immersive game worlds. Furthermore, they can hold the declarative and generic power to empower designers to author adaptive generation in a large domain of different games.

These challenges open up a variety of promising research directions. Pursuing them will lead us to the development of new methods and techniques, which in turn will improve present adaptive game technology. As a result, games and simulations can become more flexible, agile and complete in the way they adapt to the player. Ultimately, a better adaptivity will foster the potential to make games and simulations even more unpredictable, effective and fun.

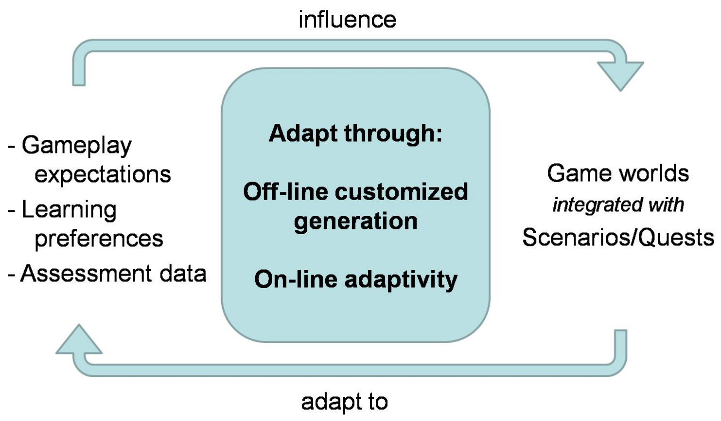


Figure 2.6: Open challenges for adaptivity in games and simulations

3 A semantic generation framework for enabling adaptive game worlds

Adaptive games are expected to improve on the pre-scripted and rigid nature of traditional games. Current research uses player and experience modeling techniques to successfully predict some gameplay adjustments players may desire. These are typically deployed to adapt AI behavior or to evolve content for simple game levels. In this chapter we propose a generation framework aimed at creating personalized content for complex and immersive game worlds. This framework captures which content provided the context for a given personal gameplay experience. This model is then used to generate content for the next predicted experience, through retrieval and recombination of *semantic gameplay descriptions*, *i.e.* mappings between content and player experience. Through its integration with existing player and experience modeling techniques, this framework aims at generating, in an emergent way, game worlds that better suit players. Dynamic game content, which responds to the player performance, has the ability to personalize player experience, potentially making games even richer, more varied and fun.

Publication notice: *The scientific content of this chapter was published in the eighth International Conference on Advances in Computer Entertainment Technology [Lopes 11a]*

3.1 Introduction

As outlined in Chapter 1, we are especially interested in the adaptation of more complex and immersive game worlds. This focus is driven by three motivations: (i) adaptive game worlds can offer new unexplored possibilities for affecting player experience, (ii) the research coupling between adaptivity and game world generation, which seems natural to us, is mostly lacking, and (iii) this coupling can allow designers to better author adaptive games.

Adaptive games typically require a two-step methodology: player modeling and content generation. This means that a coupling between both of them is not only natural, but required for adaptivity to work. One of our goals is to enable a loose coupling to existing player modeling methods, so that our generation framework can be integrated and applied in several domains.

In this chapter, we describe the conceptual scheme of a procedural generation framework for adaptive game worlds. Our aim is to generate game world content that can provide the context for personalized dynamic gameplay. Such contextualized content can be made independent of the game narrative, although compatible and coherent with it. Firstly, this solution applies naturally to games that either have simple narratives or are tightly bound to learning objectives (see Sections 3.3 and 3.4). Secondly, for more intricate plots, generation can be specified to occur within self-contained plot events, focusing on the way to achieve an outcome and not the (narrative) outcome itself.

In order to represent the personal gameplay value of game worlds, we introduce gameplay semantics, inspired by semantic game worlds, an approach that embeds the world and its objects with all information beyond their geometry [Tutenel 08]. Deploying gameplay semantics will not only allow us to steer procedural mechanisms, but also enable an interactive design of this type of adaptivity. It is therefore highly relevant to survey what we propose as the support for adaptive game world generation: virtual world semantics.

This chapter is structured as follows: in Section 3.2 we briefly survey game world semantics and position our research. In Section 3.3 our semantic generation framework is described in detail. In Section 3.4, we discuss possible scenarios where we anticipate this framework will successfully apply. We finalize with our conclusions and future work in Section 3.5.

3.2 Related work on virtual world semantics

Semantics in virtual worlds is a recent research field. Its motivations can be traced back to the early proposal by Deussen *et al.* [Deussen 98] for an ecosystem simulation model to generate an area with vegetation. Its input data, *i.e.* terrain and plant properties, and its production rules, *i.e.* space, soil and sunlight competition, raised a discussion for the need of more complete and ubiquitous information on virtual worlds and objects.

In this direction, smart objects were proposed [Kallmann 98], containing information about the possible interactions that can be executed on them. Peters *et al.* [Peters 03] took the notion of smart objects further by creating objects with information about their functionality, how NPCs can interact with them, and where important features of the object are situated. Research in Virtual Reality (VR) has also been exploring semantic representations, specifically to apply in the design of virtual environments. Latoschik *et al.* [Latoschik 05] proposed *Semantic Entities*, an object modeling method where the actions and functions of virtual objects can be specified and applied. In the same direction, De Troyer *et al.* [De Troyer 09] introduced a conceptual modeling approach for creating VR worlds, where designers can specify high-level concepts on how complex objects are composed and moved.

Our own previous work on semantic modeling explores these ideas further, by considering, in an integrated manner, both geometric constraints/relationships and functional information (we leave its detailed discussion for Section 3.3). Additionally, this semantic representation has been applied not only during runtime (as Peters' smart objects), but also to procedural content generation and layout solving (thus reassuming Deussen's requirements). We consider that it is a natural step to further integrate this approach with semantic information about the gameplay value of game worlds, and match it with player data.

3.3 Generation framework

In the previous chapter, Fig. 2.1 outlined the architectural principles typically used to support adaptive games. In essence, game logs, recording the players' performance, are used to create models of players' skills, preferences or style. Given a game state, these models can also be used to assess and predict the players desired experience of the next game state. Depending on the approach, both player and experience models can be used, in conjunction or not, to steer an adaptation and generation engine. This engine adjusts or generates the appropriate game components to better suit the player, *i.e.* adapted to the data in those models.

We are specially interested in researching within this field from an adaptation and generation perspective. Our aim is to develop a generation framework able to create game worlds that can provide the *potential* and the *context* for personalized dynamic gameplay. We use the terms *potential* and *context* since we believe it is never up to the content alone to fully realize experiences. The player and the game engine are responsible for fulfilling that potential. The goal for this framework is then to maximize the appropriateness of the generated content, to enable the fulfillment of personalized experiences.

In this section we describe our proposal for the generation framework for adaptive game worlds. See Fig. 3.1 for a conceptual scheme of the framework. It focuses on

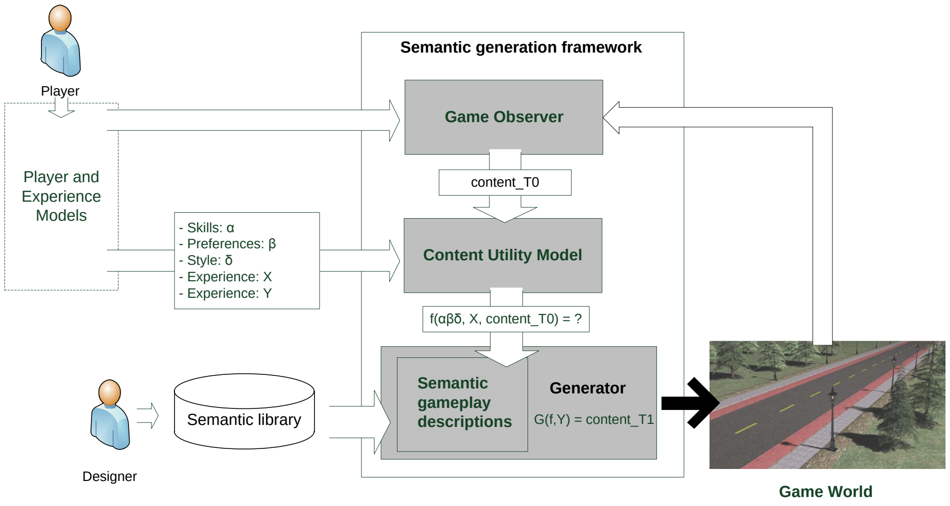


Figure 3.1: Scheme of our generation framework for adaptive game worlds: Semantic library, Semantic gameplay descriptions, Content utility model, Game observer and Generator

generating adaptive game worlds, stemming from our goal of investigating new ways of potentially affecting gameplay. As seen in Fig. 3.1, the generation process can be integrated with player and experience modeling techniques, as the ones surveyed in the previous chapter. Our aim is that this framework can be seamlessly reused with distinct player modeling methods.

With our approach, semantics encoding the gameplay value of game worlds is deployed atop geometry. This semantic information can be compared, using mechanisms similar to case-based reasoning, with outputs from the integrated player and experience modeling methods. This comparison and retrieval allows us to predict what should be the content capable of providing the next desired player experience. Using semantics we can: (i) explore the link between adaptive game worlds and procedural content generation, and (ii) consequently, enable game designers to control the generation process and author adaptivity. In the next paragraphs we will further detail our semantic generation framework.

3.3.1 Semantic library

This generation framework builds upon previous work on semantics in game worlds [Tutenel 08]. We define semantics, in the context of game worlds, as all information about the world and its objects, beyond their geometry. This includes object properties, high-level attributes and functional information, as well as interrelationships among different objects.

Each object in our game worlds typically carries all its semantics. It belongs to some class of a *semantic library* [Tutenel 09b], a hierarchical class database, partly based on the WordNet [Miller 95] ontology. This semantics can be used to control and constrain algorithmic procedures that generate specific world content. This semantic level provides designers with a powerful front-end that generates and steers an underlying procedural level, while encapsulating the complexity of the latter. This approach has already been successfully deployed, *e.g.* in interior layout solving [Tutenel 09b], procedural filters [Tutenel 11b] and building generation [Tutenel 11a], as shown in Fig. 3.2.

Designers use a library editor, *Entika* [Kessing 12], to specify semantics on each class. Two types of classes are present: entities and abstractions. Entities refer to what is possible to instantiate in a game world (*e.g.* physical objects, materials, substances). Abstractions are characteristics of entities (*e.g.* attributes, states, services) or of sets of entities (*e.g.* groups, scenes). Associating entities and abstractions allows us to specify for each game object a set of attributes, including functional information, as well as relationships with objects of other classes.

Building upon this approach, a new layer of gameplay semantics is required, so that the semantic library classes can include knowledge on how they can affect player experience. For this, we designed a set of gameplay abstractions: *player skill, player preference, player style, experience, game genre* and *actor*. Each gameplay abstraction can be associated with classes of the semantic library. The basic idea is to characterize semantic classes in terms of their gameplay value to players. Typically, a designer would create gameplay semantics by adding this type of abstractions on each class. To do so, they will use the following scheme, available for each class:

Class A:
can provide gameplay **experience(s)** Z
to players with: **skill(s)** W, **preference(s)** Y, **style(s)** X
when owned by **actor(s)** V
in **game genre(s)** U

Allowed values for Z, W, Y, X, V and U are already encoded in the semantic library.

We propose these gameplay abstractions since they naturally derive from the conclusions of Section 3.2, *i.e.* they match the main features in player and experience modeling. Also, as we will exemplify in Section 3.4, they can be further parameterized using scalar values. Each class in the semantic library can be altered to include several associations as the above. For example, a baseball bat can provide different experiences in sport or fighting games, when owned by the player or NPC.

The nature of the semantic library allows high flexibility when creating gameplay semantics. Since this can be defined for each class, both entities and abstractions can be considered. As such, gameplay semantics can be specified for a variety of different aspects, as, for example, physical objects, groups of entities or even generic attributes

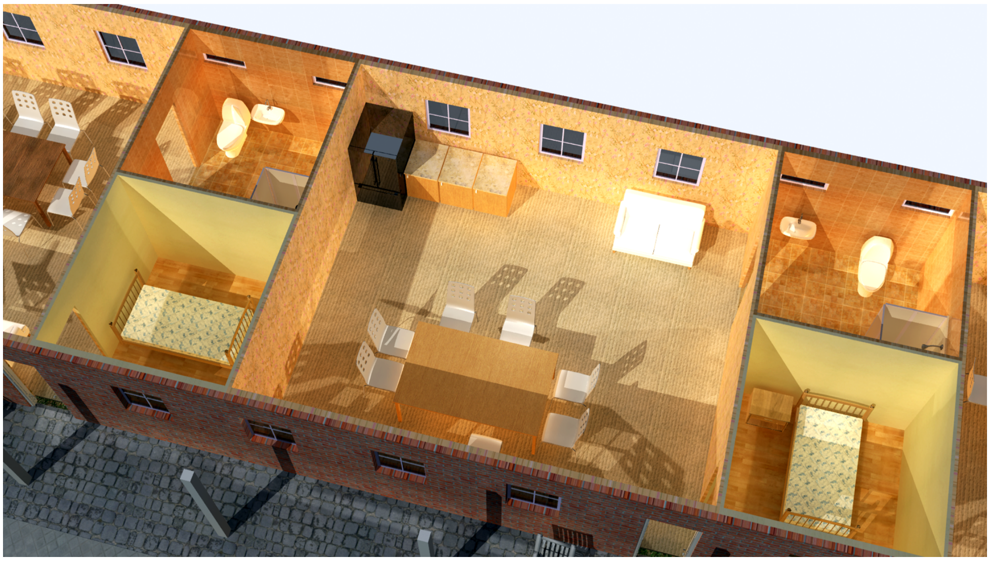


Figure 3.2: Interior layout example, generated using the semantic library [Tutenel 11a]

or states. Furthermore, the semantic library allows multiple levels of specification, where property values can be constrained or instantiated. This allows us to define and restrain gameplay semantics to different levels of class property values (*e.g.* to all sizes of an object vs. to a specific size).

3.3.2 Semantic gameplay descriptions

As we described in the previous paragraphs, and as illustrated in Fig. 3.1, game designers create gameplay semantics atop a game world semantic library. Our generation framework accesses this information through *semantic gameplay descriptions*, containers of links between semantic classes (*i.e.* game world content) and player experience. The aim of semantic gameplay descriptions is to compare them with the observed behavior and experience of a particular player, at stages that require content generation. An example of semantic gameplay descriptions is shown in Fig 3.3.

Semantic gameplay descriptions are partly inspired on case-based reasoning. They are meant to encode valid combinations between content and the gameplay experiences they can provide, for a given set of preconditions. In our case, these preconditions relate to player features (*e.g.* $\alpha\delta$ in Fig. 3.3) and game genres (*e.g.* G1 in Fig. 3.3). Semantic gameplay descriptions result from the designer-specified semantic library (see Fig. 3.1) at design stage, for each new adaptive game. The gameplay semantics of each library class is analyzed and semantic descriptions are created and assembled accordingly. The logic is simple: analyze library classes, identify

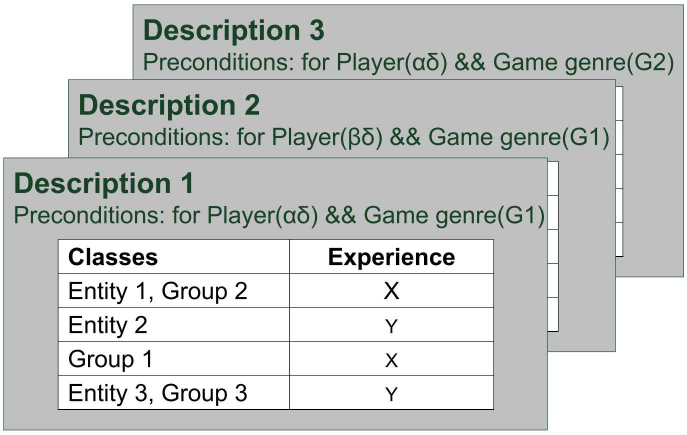

Description 3
Preconditions: for Player(αδ) && Game genre(G2)
Description 2
Preconditions: for Player(βδ) && Game genre(G1)
Description 1
Preconditions: for Player(αδ) && Game genre(G1)
Classes
Experience
Entity 1, Group 2
X
Entity 2
Y
Group 1
X
Entity 3, Group 3
Y

Figure 3.3: Example of semantic gameplay descriptions, with notation consistent with Fig. 3.1

and aggregate preconditions, create respective gameplay descriptions and add the applicable classes and respective experience to them. The accuracy of semantic gameplay descriptions is therefore dependent on the knowledge specified by game designers.

Any class from the semantic library can potentially be used in a gameplay description (*e.g.* objects, scenes, groups) not only with different levels of instantiation (*i.e.* with attribute values already specified, constrained or not) but also linked to different ways of creating its geometry (*e.g.* geometric models, procedures).

3.3.3 Content utility model

To be able to compare semantic gameplay descriptions with player behavior and experience, we need to not only model these, but also the content that enabled and provided the two aspects.

For this, we introduce *content utility modeling*, supported by two steps: (i) integration with player and/or experience modeling and (ii) monitoring the relevant content which enabled player behavior/experience. As discussed in the previous chapter, player and/or experience models typically include player behavior features and experience features, not only describing which experience was observed but also which should be the next. Their integration is achieved by a model translator, a component which converts the format of the player/experience models' output to the format of our semantic player/experience features. If complex experience modeling is not needed for integration (for example, for difficulty level adjustments), the model translator can be built to encode the behavior (even if static) of the model left out.

Content utility models are responsible for relating player/experience features with the relevant content that enabled them. To do so, we propose a virtual game observer that monitors and selects content appropriate to incorporate in the content utility model. It is critical that this observer only selects content which, with a high degree of confidence, contributed for the conclusions from the player/experience models. Otherwise, our new model would include misleading data.

As seen in Fig. 3.1, the game observer accomplishes this by monitoring both the game world and the integrated models. For this to be possible, we devised a set of criteria for selecting relevant content, which spans both domains. Regarding the game world, the observation criteria account for content that was interacted by: players, NPCs and the game engine. The nature of these interactions accounts for the use of objects and the occupation of spaces, in the same time interval as what was processed by the integrated models. Regarding the integrated models, a single criteria accounts for the content that is directly related with the game metrics they use. The idea is to observe which game metrics, measured by the model, reflect content interaction and include the content related to those metrics. For example, if a player modeling method monitors the number of walls broken by the player, the game observer should monitor and include broken walls in the content utility model. The game observer needs to be customized for every instance of a new game or a new integrated modeling technique, so that these criteria are totally fulfilled.

3.3.4 Generator

The generator in Fig. 3.1 is responsible for the creation of game worlds, by applying the information modeled in our framework. The basic approach, analogous to case-based reasoning, tries to match the content utility model with gameplay descriptions. The plausible assumptions are that: (i) if gameplay descriptions include valid content-experience combinations, and (ii) if the content utility model associates content, player features, and experience, all measured for a given moment, and (iii) if a semantic gameplay description exists with that same association, then (iv) the remaining combinations in that description can also be considered valid and usable.

Under these assumptions, generation becomes a matter of retrieval and recombination of semantic gameplay descriptions. As illustrated in the example of Fig. 3.1, if the content utility model includes *content_T0* which enabled experience *X*, for a player modeled as $\alpha\beta\delta$, then the generator needs to implement the following function *G*, where Y is the experience required for the next stage:

$$G(f(\{\alpha\beta\delta\}, X, content\_T0), Y) = content\_T1 \tag{3.1}$$

Function *f* needs to retrieve and return a set of semantic gameplay descriptions where its arguments are valid. Afterwards, function *G* examines the retrieved semantic gameplay descriptions and selects the content (*content_T1*) which provides the required experience Y. The frequency and moments on which the generator, *G(x)*, should act in a game world will need to be decided when integrating our framework in each particular game.

If only a single description is retrieved by *f(x)*, the generator has a simple task in creating the relevant content. Content from the description is selected and instantiated by *G(x)* according to the description semantics and using the associated geometric models or generation procedures. The generator will include a procedural mechanism, based on semantic layout solving [Tutenel 09b], responsible for combining the selected instances into game world sections.

If several descriptions are retrieved by *f(x)*, then the generator has a more emergent behavior. In this case it will recombine several instances of content declared in independent descriptions. This way, the content to generate emerges from distinct descriptions, maximizing the potential for a richer and more complex variability.

The variability in this emergent recombination mechanism is possible in two ways, both implemented in *G(x)*: *precondition selection* and *content selection*. For the former, we consider as valid all gameplay descriptions with preconditions which are a subset of the player modeled features. For example, in Fig. 3.1, the player was modeled as $\alpha\beta\delta$ and, in Fig. 3.3 there are descriptions for players modeled as $\alpha\delta$ and $\beta\delta$. Function *f(x)* will consider these as retrievable and selectable in *G(x)*. To solve possible conflicts between content of the retrieved descriptions, they are ordered by degree of similarity with the player modeled features.

As for content selection, increasing the number of retrieved gameplay descriptions can lead to redundant or conflicted content. So, solving these situations differently can lead to different generated worlds. Besides the degree of similarity described above, we use and compare semantics to identify and remove redundant or conflicting content, also introducing randomness in this process.

Finally, a remark on "starvation" situations where the function *f(x)* does not retrieve any semantic gameplay description for a given step: even though we do not envision such a use for our generation framework, acceptable solutions can be found, *e.g.* using predefined static content, as done with non-adaptive games, in these steps.

3.4 Application scenarios

In the previous section, we described our proposal for a semantic generation framework for adaptive game worlds. In this chapter, we aim at presenting and discussing the framework's conceptual scheme; its implementation and validation will be done in the remaining chapters of this thesis. Instead, we will describe here two simulated scenarios of the application of our framework, in order to help us evidence and discuss its advantages. Each scenario will focus on one game world section where generation is required, using what was modeled (player, experience and content) in the previous world section. Please refer to Fig. 3.1 for the framework modules mentioned throughout this section.

3.4.1 Scenario 1

The first scenario occurs in a First Person Shooter (FPS) game, where generation is required to occur between one room and the next one. Player and experience modeling are deployed to capture style and affective experience (using, for example, [Ramirez-Cano 10] and [Pedersen 10]).

Two different players have been modeled as (i) explorer, *i.e.* likes to explore the environment, and (ii) achiever, *i.e.* likes collecting items and leveling up his abilities through balanced gameplay. Both were modeled as being bored and needing a 50% increase in excitement. The content utility model registers the following content, for both players: previous room had one division, four empty boxes were opened, three crates were used as hideouts for NPC and a time-bomb was activated by the player.

For the explorer player, a single gameplay description is retrieved stating that rooms with one division, zero hidden chambers and time bombs increase boredom. This gameplay description also declares that hidden chambers in rooms increase excitement in 10%. From this, the generator decides to create the next room with five hidden chambers.

For the achiever player, a single gameplay description is retrieved where empty item boxes and NPC-owned crates increase boredom. This gameplay description also states that leveling up items that increase player abilities increase excitement in 25%. So the generator instantiates two leveling up items in the next room.

Discussion

In the scenario above, we can highlight the flexibility of the semantic library in allowing designers to specify how content can influence player experience. Boredom for the explorer is characterized by simple one-division rooms and by time bombs. This way, gameplay semantics can be used to capture that simpler rooms leave no space to explore and time bombs, if activated, can leave no time to calmly explore. The same authoring expressive power is shown for the achiever, where empty item boxes and NPC-owned crates increase boredom. This expresses the designer's knowledge on the fact that achievers like to progress in the game, by having multiple opportunities of collecting items, in a balanced way, where advantages by NPC are disliked.

The potential of our generator is also demonstrated in this scenario. In the end, two players with different styles were given two different rooms, with content more appropriated for each one. This shows the value in considering game world content in adaptive games. Also, in this scenario, the main gameplay was not hindered. The type of generated content was specified to not relate with the main goals or the mission of the game; it affects the player experience in that specific moment and room. So, although somewhat different, the game still remains the same for both players. This scenario also highlights the ability of our generator to function in a mixed mode: the room could actually be the same geometric space with some static content, and only some key objects are generated.

3.4.2 Scenario 2

The second scenario occurs in a driving simulator, a serious game where a player drives around a city and executes what an instructor suggests. Generation occurs on each street. Player modeling is integrated to capture the skills of the player, and experience is modeled to encourage, in an transparent way, the development of the captured skills (as, for example, in [Westra 10]).

For a player, skills are modeled as: 0.4 (out of 1) proficiency in parking sideways and 0.6 in clutch balancing. In this case, the modeled experiences are redundant and directly mapped from skills, *i.e.* low and medium proficiency in, respectively, parking sideways and clutch balancing (here defined as features *a* and *b*). Consequently, the learning goals to be encouraged next are thus to improve on these skills, using different learning levels, with an accessible level on parking sideways (*c*) and a challenging level on clutch balancing (*d*), both due to the measured skill levels. Besides these player and experience features, the content utility model includes: a one lane steep road *(e)*, misplaced parked cars on the side of the road *(f)* and traffic lights placed on a steep road *(g)*.

Two gameplay descriptions are retrieved, for the two types of modeled player skills. Tables 3.1 and 3.2 describe the content of these descriptions. We signaled both the classes and the features which match what was captured in the content utility model, *i.e.* arguments *X*, *content_T0* and *Y* of the *G(x)* function, as explained in Section 3.3.

Table 3.1: For players with proficiency ≤ 0.5 in parking sideways

Semantic class	Experience
e) Road(lanes=1)	*a)* low proficiency: parking sideways
f) Car(parked=misplaced)	*a)* low proficiency: parking sideways
z) Road(lanes≥2 , steepness=0)	*c)* accessible level: parking sideways
y) Car(parked=parking space)	*c)* accessible level: parking sideways

Table 3.2: For players with proficiency ≥ 0.5 in clutch balancing

Semantic class	Experience
e) Road(steepness=30)	*b)* medium proficiency: clutch balancing
g) Stoplight(location=*e*,wait=40)	*b)* medium proficiency: clutch balancing
x) Road(steepness=45)	*d)* challenging level: clutch balancing
w) Stoplight(location=*e*,wait=60)	*d)* challenging level: clutch balancing

Since both descriptions are retrieved, their entries are combined, generating a road with two lanes (*z*), with cars parked inside parking spaces (*y*), traffic lights with longer waiting periods (*w*) and with steepness angle 45 (*x*). The dilemma between a leveled or a steep road (*z* and *x*) would be solved by comparing skill proficiency.

The parking sideways skill is closer to be matched (0.4 from the model to 0.5 from the description) than the clutch balancing skill (0.6 from the model to 1 from the description), and could therefore be considered less important to be encouraged.

Discussion

This scenario highlights the authoring flexibility of our framework, showing its applicability in a different context: skill-based learning. Here, gameplay semantics can still be created to evidence the need to respond to different players by using different content. For example, for parking sideways, more street lanes lead to more space and thus less pressure, leveled streets mean more visibility and cars inside parking spaces lead to more space to maneuver. In other words, using the semantic library, designers can specify that dissimilar players can benefit differently from different content.

Scenario 2 also highlights the emergent behavior of the generator. The final scene is created through the combination of two retrieved gameplay descriptions. This can increase the variability in the generated content, in an unique way. In this scenario, this is made possible by considering two player experience requirements simultaneously, a usual behavior in present adaptive games. The conflict resolution in this example is based on a similarity degree between the models and the descriptions, together with a criterion to focus on the skill needing more improvement. We foresee that mechanisms like these should be directly specified in the generator. This scenario also highlights the ability of our generator to act in a fully procedural mode, where all content is generated.

Although not directly mentioned in these scenarios, we should also discuss performance and burden on designers. This framework is envisioned to perform on-line, while the game is running. Considering that each semantic class should be already instantiated as of the import of gameplay descriptions (*i.e.* at the design stage), the remaining bottlenecks are likely the retrieval of descriptions and the layout solving by the generator, both at run-time. Through the experiments on the remaining chapters of this thesis, we verified that an indexing mechanism for descriptions was enough to achieve satisfactory generation efficiency.

As for the process of creating gameplay semantics, it needs to avoid burdening designers with overwhelming manual effort. The semantic library already provides mechanisms to facilitate the specification of semantics, including control on class inheritance and automatic consistency checks.

3.5 Conclusions

In this chapter, we presented the conceptual scheme of our semantic generation framework for adaptive games. We have discussed its main components, highlighting the role that semantic modeling can play in adaptive game worlds. Through this framework and, specifically, by means of gameplay semantics, game worlds are generated to match integrated player and experience models, *i.e.* adapting to the players' goals and needs. Semantics about personal gameplay value, associated with game world geometry, steer a procedurally-based generator in creating a more suitable context for personalized dynamic gameplay.

The novelty and advantages of this framework can already be highlighted here. First, this framework allows us to include 3D immersive game worlds among those game components targeted by adaptivity. Achieving this, in turn, can trigger investigation on new, unexplored ways of affecting gameplay.

Second, regarding the process of creating adaptive games, the framework can allow content generation methods to be loosely coupled with a variety of player and experience modeling.

Third, the framework brings about the first inclusion of gameplay information in the area of semantic modeling. Although it is here being used for the procedural generation of game worlds, in the future, this new semantic scheme will likely become valuable, for example, for runtime interaction with game objects.

Finally, we can expect that deploying more and richer semantics will enable game designers to much better author adaptivity.

4

Using gameplay semantics to procedurally generate player-matching game worlds

The use of procedural content generation to support adaptive games is starting to gain momentum in current research. However, there are still many open issues to tackle, namely the control over such generative methods. Our research focuses on authoring adaptivity in games, *i.e.* controlling reusable and generic methods for linking the procedural generation of 3D game worlds with gameplay, as measured by player modelling techniques. As the interface for that link, we propose the use of *gameplay semantics*, a knowledge representation technique that allows our generator to match content to player models. We present and discuss the implementation of our proposed method in an existing racing stunt game. Gameplay semantics is created by designers in a generic way and is then used to procedurally generate player-matching game worlds, both at design and at game stage. Current results show that our approach can automatically create such adaptive game content, thus effectively bridging game world designers, procedural generation and gameplay.

Publication notice: *The scientific content of this chapter was published in the third workshop on Procedural Content Generation in games* [Lopes 12]

4.1 Introduction

In the previous chapter, we proposed the use of *gameplay semantics* as the interface to match the player behavior and experience with the game content generator [Lopes 11a]. With this semantics, *i.e.* embedded gameplay knowledge about virtual world objects beyond their geometry, our framework can support the generation of player-matching game worlds, in games where this is applicable and valuable.

This chapter investigates the suitability and power of this approach, discussing our implementation results in a modification of an existing game, *Stunt Playground* [1]. We explain how our semantic approach can be easily applied to create a game with player-matching game worlds, in an off-line fashion (on-line generation will be approached in the next chapters). This is achieved by integrating our semantic framework with a generation process for game worlds and with player and experience modeling techniques. In this case, we added semantic layout solving techniques for generating *Stunt Playground* game worlds and also created new player and experience models. We showcase our results, *i.e.* game worlds generated at game stage, discussing the added value of the new adaptive *Stunt Playground*.

This chapter is structured as follows: in Section 4.2, we give a brief overview of our previously proposed semantic generation framework and highlight the new contributions for the actual generation process. In Section 4.3 we explain the integration of our methods with *Stunt Playground* and the player modeling and content correlation methods. Section 4.4 presents and discusses our results, preceding our conclusions, in Section 4.5.

4.2 Semantics and generation

In the previous chapter, we outlined our proposal for a framework aimed at generating player-matching content for complex and immersive game worlds. In this section we will first summarize this framework, to give context to the rest of this chapter and we will also outline new contributions on the generation process (layout solving), not presented before.

4.2.1 Semantic framework

Our semantic generation framework (see Fig. 3.1) is responsible for integrating behavior and experience modeling with procedural content generation, possible through the use of game world semantics.

Designers use the semantic library to deploy information on several semantic classes (or entities), *e.g.* objects, attributes or groups. Gameplay semantics is one of the many layers of information that can be deployed. It describes the gameplay

[1]Stunt Playground was developed by Tim FitzRandolph using the Ogre3D engine. Available as freeware at: http://walaber.com

value of various entities and can be specified in terms of: (i) gameplay experiences, (ii) player behavior features and (iii) involved game actors. For example, a designer can specify that a car ramp (the semantic entity) entails a certain level of fun (experience) for a reckless driver (behavior), when used by the player (actor) in a racing game. Gameplay semantics is defined by designers, and can be as generic and reusable as the designer wants them to be.

Semantic gameplay descriptions are automatically converted from the gameplay semantics layer in the library and are partly inspired by case-based reasoning. They are knowledge containers which encode valid combinations between semantic entities (the car ramp) and player experiences (fun provided). They are individually organized by the case preconditions each one applies to, *i.e.* the player behavior features (*e.g.* reckless driver). This way, adaptive content generation becomes a retrieval and recombination problem where the solution is to find the most appropriate semantic gameplay description, *i.e.* the best case, for a given moment.

The input of such a retrieval process is given by the behavior and experience models, integrated with the framework. Behavior models supply the behavior features (a certain level of reckless driving) to retrieve the matching gameplay descriptions. Experience models help retrieval by supplying not only the next expected gameplay experience (a certain level of fun), but also the previous measured gameplay experience (*e.g.* player was not having fun). Additionally, we developed a game observer component to further help the retrieval process. It correlates which game content enabled the previous gameplay experience. As explained in the previous chapter, the assumption is that if a semantic gameplay description includes an *observed* content-experience association, then the remaining associations can be considered valid to be used.

In our framework, the retrieval of multiple descriptions allows this generation process to be emergent. For a description to be selected, it is enough to include only one (and not all) case precondition matching an input behavior feature. This means that finding multiple descriptions and combining their retrieved content is possible and increases variability beyond what designers declared. Solving retrieved content is up to the post-retrieval generator.

4.2.2 Post-retrieval content generation

After retrieval, it is still necessary to synthesize such player-matching content into a meaningful game world (segment). The post-retrieval generation we describe here is responsible for this.

For post-retrieval generation, we make use of previous work on semantic layout solving (see [Tutenel 09b] for details). This choice was motivated by two reasons: (i) the input of the semantic layout solver fits very well with the output of the retrieval generation process, and (ii) the solver is already fully integrated with the semantic library. The use of other generators should be possible, as long as the generator can synthesize large scenes from individual content.

Given an initial layout, the semantic layout solver can stochastically position individual objects in that layout, complying with a set of rules. These placement rules take into account the relationships, rules and predicates defined in the semantic library. Each entity in the library includes a semantic representation consisting of *object features*, *i.e.* tagged 3D shapes. The object features we used are *Clearance* and *OffLimits*. A *Clearance* feature cannot overlap with any other features except for other *Clearance* features (shared open areas) and an *OffLimit* feature cannot overlap with any other feature type (solid areas). Additionally, we used object features *back*, *front*, *top*, *bottom*, *left* and *right*, which define the respective 3D shape of each of those object boundaries. These features are used in placement rules to derive valid locations and orientations for each entity in any layout. The layout solver includes rules for *against*, *around* and *on*. As an example, if the semantic entity *plate* has an *on* rule with the feature *top* of the semantic entity *table*, the solver places the plate somewhere on the table top.

Each placement rule requires its own individual procedure, within the layout solver, to calculate all valid locations. These procedures are the geometric realization of the respective placement rule and return a group of multiple possible locations. As an example, the individual procedure for the *on* rule uses the Minkowski subtraction between both features of both semantic entities to calculate the polygonal shape that contains all possible valid locations. The layout solver is also responsible for selecting the best order in which to solve placement rules. The order to pick entities to place is defined by a dependency graph, with the following sorted criteria : (i) least outgoing rules to entities not yet picked, (ii) most incoming rules, (iii) most outgoing rules to already picked entities, (iv) largest entity. If semantics are correctly defined in the library, then they, together with the ordering algorithm, are enough to solve layouts, and if there are no available valid locations left, an entity is simply not placed.

4.2.3 Contributions to semantic layout solving

Regarding the semantic layout solver of [Tutenel 09a], the existing object features and placement rules, as defined by designers, are the basis for post-retrieval generation to work. The retrieval process feeds its result, *i.e.* the player-matching content, to the solver that, supported by the semantics of the library, creates a valid game world.

However, upon analysis of the layout solver of [Tutenel 09a], we identified additional placement rules and object features necessary for player-matching game world generation. As a result, these were created and added to the semantic layout solver.

The new *empty when placed* object feature defines a shape which cannot overlap with any other object feature, but only at the moment when the entity possessing this feature is being placed. Afterwards, that shape can overlap with any entity or feature. This feature is typically used to express a precondition to a placement rule with another entity. For example, if a ramp should be placed before some obstacle, then the placement of the ramp should consider and reserve some empty space in front of it for future occupation by an obstacle. The implementation of such a new feature builds upon the *Clearance* and *OffLimits* features. For each placement rule and its two semantic entities, a Minkowski sum is used to calculate an area containing all locations for which the *empty when placed* shape would overlap with the already present shapes (objects, *Clearance*, *OffLimits*, etc). This area is then subtracted from the possible placement locations.

The new *follow on* placement rule constrains the possible locations of a source and a target entity. This rule is used to specify that a source entity should be placed in a layout, aligned with an already placed target entity, *i.e.* where both form a trajectory. For example, an obstacle should be placed following on an already placed car ramp, in a certain trajectory. This new rule implied the creation of new additional features: *mid-axis x, y* and *z*. These features define a plane shape which cuts an object in half, in each of the 3D coordinates. For flexibility purposes, we also added similar spaces for cuts in quarters. The new and existing features can be used to define the trajectory of the source and target entities.

The implementation of this new rule involved a new dedicated procedure. For a target feature (*e.g.* center line of a car ramp), a new extruded line is defined by calculating two incremental points (the incremental range is either defined from designers parameters or, if not present, randomly) in the same direction as the feature. The center point of the source entity is then randomly placed somewhere on that line, rotated to respect the source feature plus an optional rotation parameter. Even though this *follow on* procedure constrains the placement of the source entity, there are still some degrees of randomness and, thus, variability: the initial placement of the target entity, the range of the placement line (if not defined by designers) and the selected point in the placement line.

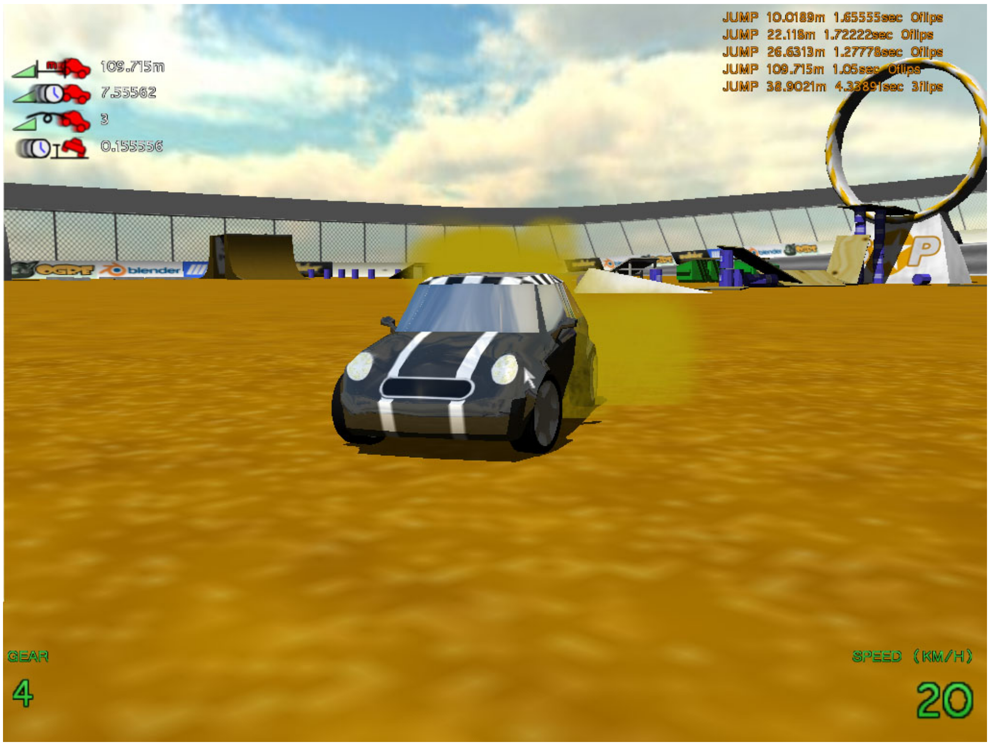
4.3 Player-matching game worlds

This chapter describes the integration of our semantic generation framework into an adaptive game. In this section we will outline its implementation and research issues.

We chose to integrate our approach with an existing game, in order to assess how generic and applicable the semantic generation framework is. The chosen game was *Stunt Playground*; see Fig. 4.1 [2]. In this single-player game, players can drive around, free to do stunts in an arena with a variety of props. It is an open sandbox game with no scoring, progression or goals. It includes a game world editor, where the user can create, save, load and play arenas, apart from the ones already included in the game.

For this research, we developed and added a new adaptive game mode, in which the player starts in a predefined initial arena. After playing a game cycle of five minutes, a new arena is generated each time to better fit the player's style. Unique arenas are generated every game cycle for every player. This new mode entailed

[2]Stunt Playground was developed by Tim FitzRandolph using the Ogre3D engine. Available as freeware at: http://walaber.com



```
JUMP 10.0189m 1.65555sec 0flips
JUMP 22.118m 1.72222sec 0flips
JUMP 26.6313m 1.27778sec 0flips
JUMP 109.715m 1.05sec 0flips
JUMP 38.9021m 4.33891sec 3flips
```

Figure 4.1: Stunt Playground gameplay

the creation of methods for player and experience modeling, content correlation, gameplay semantics and procedural content generation, which will now be described.

4.3.1 Behavior and experience modeling

As stated in Section 4.2, our framework needs to integrate behavior and experience modeling, in order to generate player-matching game worlds. For Stunt Playground, we developed our own behavior and experience models. Due to the simplicity of the game mechanics, we opted for heuristic vector-based models, which seem to be effective enough, as demonstrated by Westra *et al.* [Westra 10]. All the heuristics described below, for both models, were the result of empirically observing various informal game sessions with several player types.

For modeling player behavior, we defined two simultaneous scales able to capture a stunt driver's playing style. The *Evel Knievel* scale measures how much of a reckless stunt driver a player is. On the other side, the *Sunday Driver* scale measures how much cautious a player is in performing stunts. We used two simultaneous measurements because we aimed to capture overlaps between one behavior and the other. Our initial idea was to investigate the existence of a dual complex behavior where a player might drive around acting as cautious and reckless simultaneously. For example, using only one measurement would be less expressive in characterizing a player who drives very fast, but does not interact with any prop or does any jump.

For both measurements, we use the following heuristics, which are logged and measured at run-time, and stored in a vector:

- i : ratio between the distance spent in the air (jumps) and the total driven distance;
- j : ratio between the time spent in the air (jumps) and the total game time;
- k : ratio between the measured average speed and the maximum possible speed;
- l : ratio between the number of flips which were made, and a maximum number of flips.

Heuristic k is particularly interesting since it hides a fifth measurement. On each arena, the player can choose from a group of vehicles with different maximum speeds, ranging from a bus to a racing car. However, the maximum speed in heuristic k is fixed to the maximum speed achieved by the fastest vehicle. This way, this heuristic indirectly reflects the impact of choosing a faster or slower vehicle, an important factor in characterizing a player. As for the maximum number of flips in heuristic l, they are calculated proportionally to the props available in the arena, with an average of flips per prop per time experimentally determined.

The following formulas define the final values calculated at the end of each game cycle, and determine, respectively, the measurements for the *Evel Knievel* and *Sunday Driver* scales:

$$EK = \frac{\frac{(\sqrt{i}-\tilde{d})}{(1-\tilde{d})} + \frac{(\sqrt{j}-\tilde{t})}{(1-\tilde{t})} + \frac{(\sqrt{k}-\tilde{s})}{(1-\tilde{s})} + 2\left(\frac{(\sqrt[3]{l}-\tilde{f})}{(1-\tilde{f})}\right)}{5}$$

$$SD = \frac{\sqrt{1-\frac{i}{\tilde{d}}} + \sqrt{1-\frac{j}{\tilde{t}}} + \sqrt{1-\frac{k}{\tilde{s}}} + 2\left(\sqrt[3]{1-\frac{l}{\tilde{f}}}\right)}{5}$$

where $\tilde{d}, \tilde{t}, \tilde{s}, \tilde{f}$ represent, respectively, reference values for ratios i, j, k and l, expected for the ideal neutral player and dependent on the props in the arena. Thus, these formulas capture the normalized average deviation from the experimentally determined neutral behavior. We choose to apply quadratic functions to each heuristic to better capture the faster growth rate observed for each measurement. We applied a cubic function and weight 2 to heuristic l to give additional importance to the number of flips, a maneuver which is both harder to perform and a better indicator of recklessness.

Due to the pure sandbox nature of Stunt Playground, we decided that the adaptation goal would be the maximization of the game fun factor, in every scenario. Although fun is a complex concept, virtually impossible to accurately measure, the limited nature and goal of Stunt Playground's game mechanics encouraged us to simplify it to a more attainable level. Therefore, for modeling the player experience we defined a simplistic measurement for capturing the fun experienced by players. For this, we used the following heuristics:

- m : initial fun value, for finalizing an arena (game cycle);
- n : difference between both behavior modeling scales;
- o : number of re-spawns (if stuck somewhere or bored, the player can choose to be relocated back to the initial position);
- p : ratio between the time spent totally stopped and the total game time.

Heuristic m was fixed as half of the maximum fun value (1), to represent an average game session. Heuristic n was introduced to capture an implicit goal of gameplay: to experience the progression towards a gradual specialization in one preferred type of behavior (in this case recklessness vs. cautiousness). The calculation of the fun factor in the end of each game cycle is easily explained: heuristics m and n are added, and heuristics o (normalized against an experimental maximum reference value) and p are subtracted from that value. These factors are weighted to reflect the experimentally observed relative importance of each heuristic. Additionally, the result is normalized between 0 and 1.

Validation of both the behavior and experience models was performed empirically by both observing game sessions and informal questionnaires. Both models were initially considered valid and expressive in capturing the intended information.

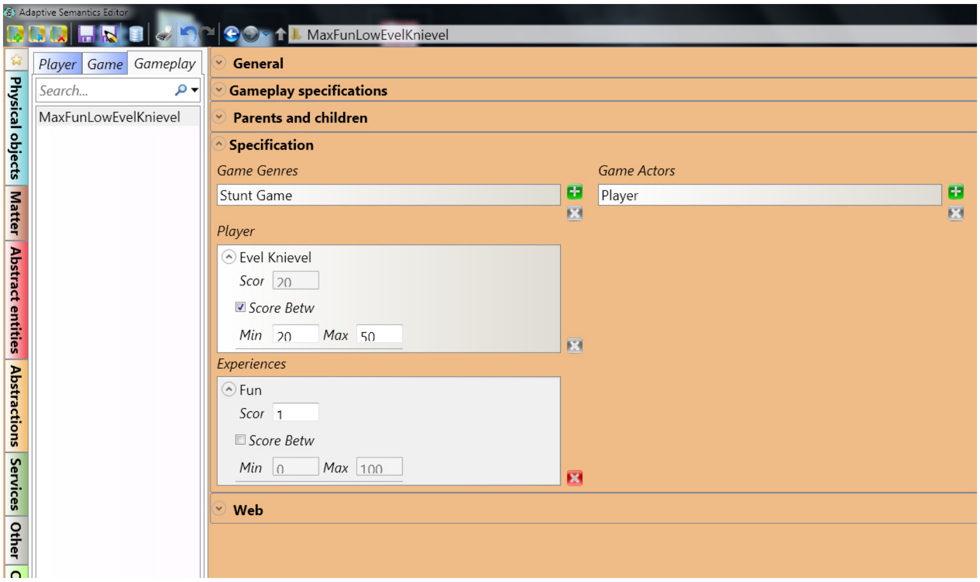
4.3.2 Correlation with content

As illustrated in Fig. 3.1, the measured behavior and gameplay experiences need to be correlated with the game content that enabled them. These correlations are used as inputs for the retrieval of semantic gameplay descriptions. In the previous chapter, we proposed a set of criteria to be implemented by a game observer. Our hypothesis is that these criteria can be sufficient to correctly establish those correlations.

In Stunt Playground's case, the implementation of the game observer's criteria was straightforward. The simple nature of the game mechanics, and the behavior and experience models explained above, both show that the content interacted by players is enough to account for the correlations. Therefore, the Stunt Playground game observer keeps track of the arena props a player interacts with and correlates that list with the behavior and experience models' final values.

4.3.3 Gameplay semantics

Gameplay semantics is defined by designers, using a semantic library editor for that purpose. Figs. 4.2 and 4.3 illustrate a screenshot of the semantic library editor, *Entika* [Kessing 12]. For Stunt Playground, we defined all the required semantics, not only for the gameplay semantics layer, but also all remaining necessary knowledge (*e.g.* placement rules and features for semantic layout solving). Although semantics is specified manually, there are some mechanisms in place to prevent this task from becoming too tedious. The use of WordNet, inheritance between semantic entities or the creation of stand-alone gameplay semantics (to then re-apply to different entities faster) are examples of this.



MaxFunLowEvelKnievel

Game Genres: Stunt Game
Game Actors: Player

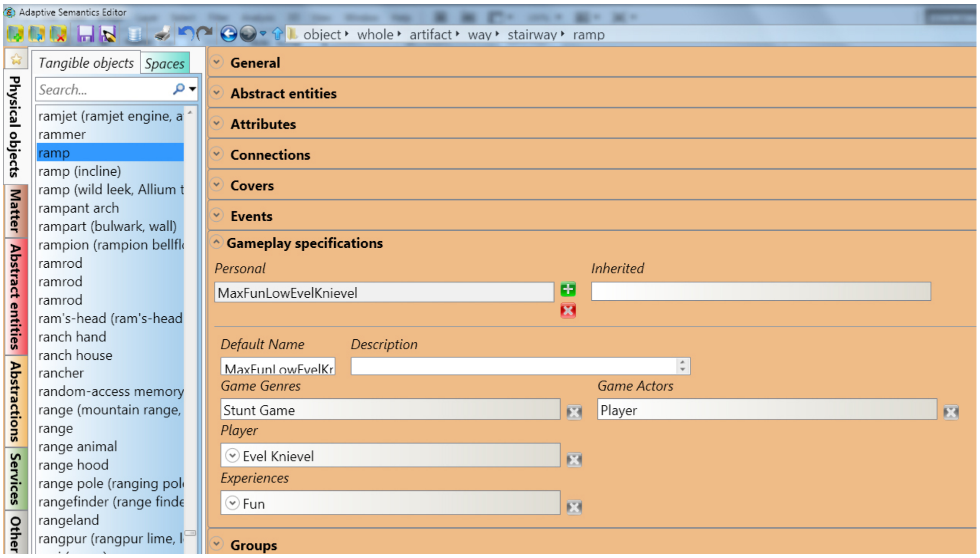
Player
Evel Knievel
Scor 20
☑ Score Betw
Min 20 Max 50

Experiences
Fun
Scor 1
☐ Score Betw
Min 0 Max 100

Figure 4.2: The semantic library allows associating player behavior (Evel Knievel) and experience (Fun) features to create and save gameplay semantics blocks (MaxFunLowEvelKnievel)

For this case study, we created all semantics before designing the player models. The goal was to keep semantics independent from the modeling methods, able to capture all the knowledge the semantic library offers. The only constraint was the mandatory integration requirement: gameplay semantics should use the same vocabulary as what is being modeled. In this case, the player styles for *Evel Knievel* and *Sunday Driver* and a *Fun* parameter.

As explained, we decided that the goal of adaptation would always be to maximize fun, measured as described above (see Section 4.3.1). This meant to apply a Fun parameter, at maximum value (100% or 1), to different semantic entities, depending



Personal: MaxFunLowEvelKnievel
Inherited:
Default Name: MaxFunLowEvelKr
Description:
Game Genres: Stunt Game
Game Actors: Player
Player: Evel Knievel
Experiences: Fun

Figure 4.3: Gameplay semantic blocks created in the semantic library can be applied and tag different entities. In this example, a ramp will be specified as appropriate for the player behavior and experience features of the MaxFunLowEvelKnievel gameplay semantics

on the type of player they apply to, *i.e.* a value for the *Evel Knievel* or *Sunday Driver* player styles. We also applied a lower fun value to semantic entities, depending on whether they were not deemed appropriate for the corresponding player styles. We devised this binary behavior (fun vs. not fun) due to the simplicity of Stunt Playground and the research goals for this chapter (implement, integrate and test our framework).

We created six levels of player styles: low, medium and high proficiency as *Evel Knievel*, and low, medium and high proficiency as *Sunday Driver*. Each level was quantified as ranges: 20% - 50% (low), 51% - 74% (medium) and 75% - 100% (high). These rates were defined following conversations with game designers and players.

The described gameplay semantics originated six semantic gameplay descriptions. They refer to the six independent levels of player styles and contain different correlations of semantic content and fun levels. Although these descriptions are disjoint in nature (due to the six independent player style levels), they can still be combined during retrieval. This happens because, as explained before, the input of the retrieval process always includes simultaneous player modeling measures for the *Evel Knievel* and *Sunday Driver* styles.

4.3.4 Generation

We integrated our approach with Stunt Playground (developed in C++) using both the semantic generation framework and the semantic layout solver (both developed in C#). The behavior/experience modeling and content correlation methods were developed directly in Stunt Playground's code (due to their dependence to the game's nature) and acted as input for the retrieval process. In the end of each game cycle (each arena), the models' values are used to retrieve content from the appropriate semantic gameplay descriptions. Although descriptions already include knowledge about the quantity of each semantic entity (in this case, stunt props), we also made that quantity proportional to the values of the player models. The semantic entities contained in the gameplay descriptions refer to the same content used in the game since they include the same identifiers and the same model meshes (Ogre3D models, in this case).

After content retrieval, the semantic layout solver generates a new player-matching stunt arena. The layout solver places the retrieved content within the basic input mesh of the arena, following the semantic placement rules. As explained before, Stunt Playground includes a game world editor to create, save and load stunt arenas. Since this editor uses XML as the format to represent game worlds, we are able to store all generated arenas for each player.

Concerning generation, we developed a game design prototyping tool to generate player-matching stunt arenas [3]. This tool works outside the game, at the design stage, and within a clone of the semantic library editor. Designers can input values for gameplay semantics levels (described in Section 4.3.3) and a matching arena is generated as an XML file. This arena can then be loaded in Stunt Playground and played. This prototyping environment is a valuable contribution for creating game worlds at the design stage, fitted for input player types and game experiences. It can be used both for testing purposes or to deploy worlds in a game where there are no player modeling methods available, but where the player/gameplay profile is known beforehand.

4.4 Results

Our results for this case study relate to the success in procedurally generating player-matching game worlds. In this section we present our early results, *i.e.* some examples of typical player-matching game worlds generated during gameplay. We saved all generated stunt arenas and logged all the data, both collected from players and calculated by the models. For this section, we showcase various examples that are especially representative of the global results. Table 4.1 shows which data matches with the examples chosen.

[3]Demo videos of the design tool and the game adaptation available at http://rlop.es

Table 4.1: Behavior and experience model data used as input for the generation of each game world in the corresponding example. The meaning of each parameter is described in Section 4.3.

	Fig. 4.4	Fig. 4.5	Fig. 4.6	Fig. 4.7	Fig. 4.8	Fig. 4.9
i (m/m)	18.1/1013.5	39/892.6	211.2/1206.9	780.6/1505.2	105/859.6	84.2/851.3
j (s/s)	2.3/300	10.3/300	35.6/300	69.6/300	41.3/300	29.3/300
k (kmh/kmh)	52.7/150	49.1/150	60.7/150	79.4/150	58.2/150	66.3/150
l (#flips/#props)	0/6	0/10	4/14	24/18	1/6	1/6
EvelKnievel	0	0	**0.62**	**0.75**	**0.21**	**0.2**
SundayDriver	**0.79**	**0.65**	0	0	**0.4**	**0.4**
o (#)	6	10	19	12	6	2
p (s/s)	25	18.5	22.3	6.7	15.6	9.1
Fun (before)	0.73	0.7	0.66	0.83	0.49	0.54

Fig. 4.4 and Fig. 4.5 show game worlds that were generated in response to a detected "low" fun experience of players modeled with, respectively, a high and a medium value of the *Sunday Driver* style. Each of these game worlds used different retrieved semantic gameplay descriptions, containing different semantic content. The main variations, besides the quantity of content (*e.g.* the amount of hurdles), resides in prop types. The less extreme Sunday Driver arena (Fig. 4.5) does not include cones, but adds new types of small ramps and new possible layouts for them (notice the liftoff-landing ramp layout in Fig. 4.5). In these examples, the amount of variability is limited by the number of stunt props available in the game. Of the 15 available props, we only deemed 8 as appropriate for this player type. This ends up not hindering gameplay that much, since the game is naturally biased towards encouraging players to be *Evel Knievel*. Since the goal of the game is to perform stunts, it was unusual for players to assume a Sunday Driver performance, even with less props.

Fig. 4.6 and Fig. 4.7 are examples of game worlds generated for, respectively, a medium and a high value of *Evel Knievel* player style. The variations observed between them are also explained by the retrieval of different semantic gameplay descriptions, which include different quantities and different (more complex) layout possibilities (whereas the props nature is approximately the same). Fig. 4.7 shows the typical upper limits (in terms of complexity and richness) of exclusively *Evel Knievel*-based game arenas. This is also the performance lower limit of our framework: generation took, in average, 5.1 seconds, a value acceptable while players wait between arenas. This value is mostly dependent on the performance of the semantic layout solver.

Even though low proficiency levels are not exemplified here, the progression of Fig. 4.4 through Fig. 4.7 demonstrates that our framework is able to generate game arenas that are highly dependent on the player behavior and experience. They were generated for different players, but always for the third game cycle (*i.e.* as the second

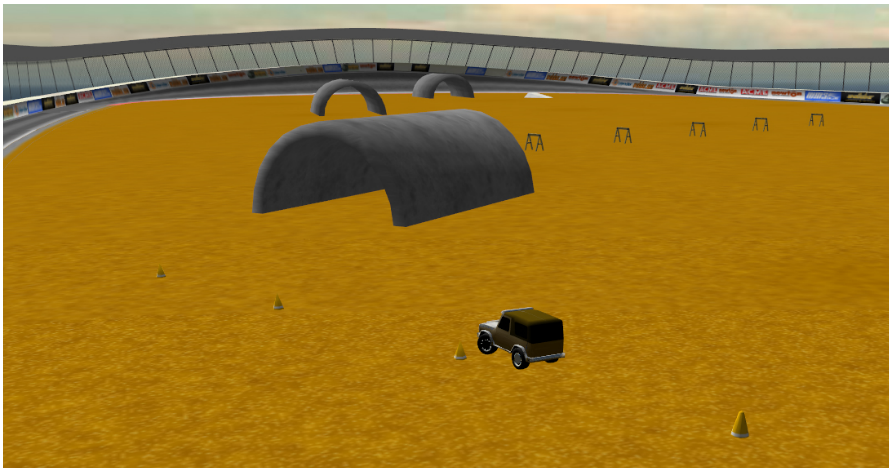


Figure 4.4: Stunt arena generated for player modeled as high *Sunday Driver*.

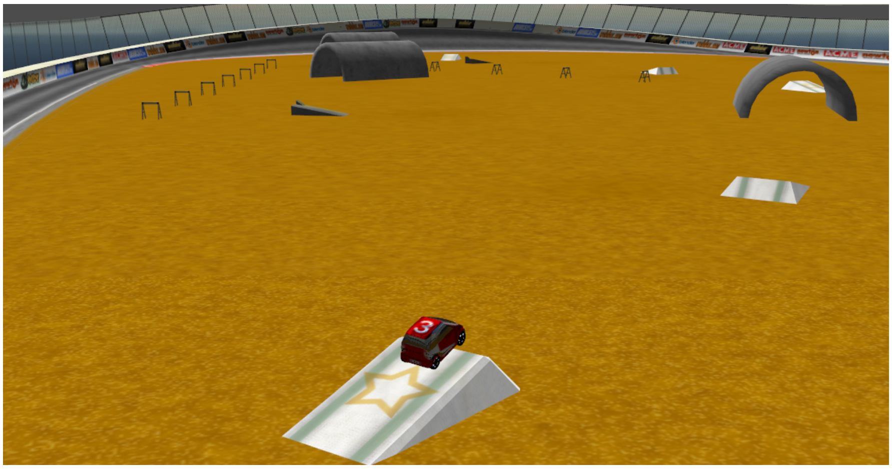


Figure 4.5: Stunt arenas generated for player modeled as medium *Sunday Driver*.

generated arena), showing that the game content at that stage is strongly adapted to how the game was played until then.

Fig. 4.8 and Fig. 4.9 illustrate arenas that were generated for players modeled as part *Evel Knievel* and *Sunday Driver*. They demonstrate the emergence in our generation framework and the variability of the generator for a similar input. The emergence is possible through the retrieval of multiple semantic gameplay descriptions, for both

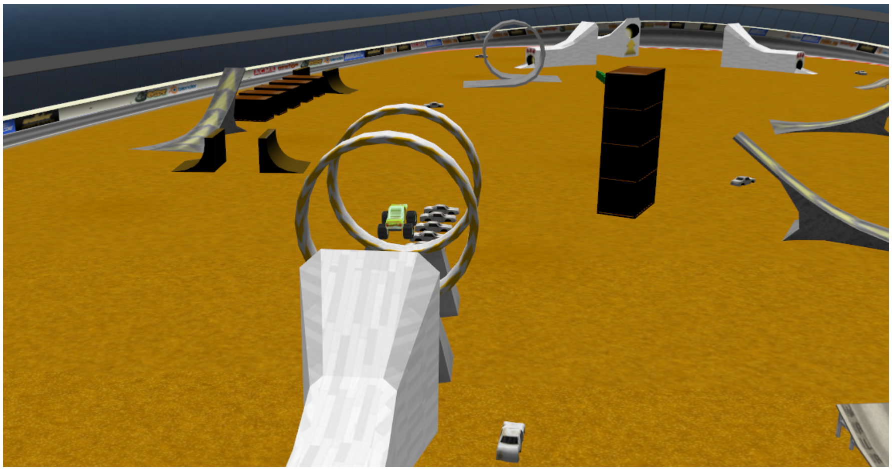


Figure 4.6: Stunt arenas generated for player modeled as medium *Evel Knievel*.

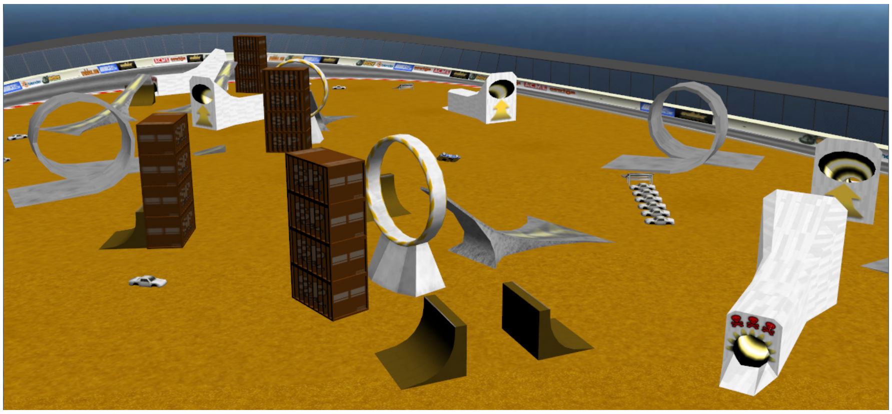


Figure 4.7: Stunt arenas generated for player modeled as high *Evel Knievel*.

Evel Knievel and *Sunday Driver*, and is illustrated by the presence of content appropriated to both player types. These arenas are emergent in a sense that the rules of their generation (*i.e.* the gameplay descriptions) were not entirely described by designers, but are derived from a combination of those. The semantic layout solver is responsible for the variability between both examples. This is illustrated not only by the quantity of props and their layout (notice the white tunnel ramps on both cases), but also the content that was actually placed, *e.g.* the tunnels of Fig. 4.8 and the hurdles of Fig. 4.9.

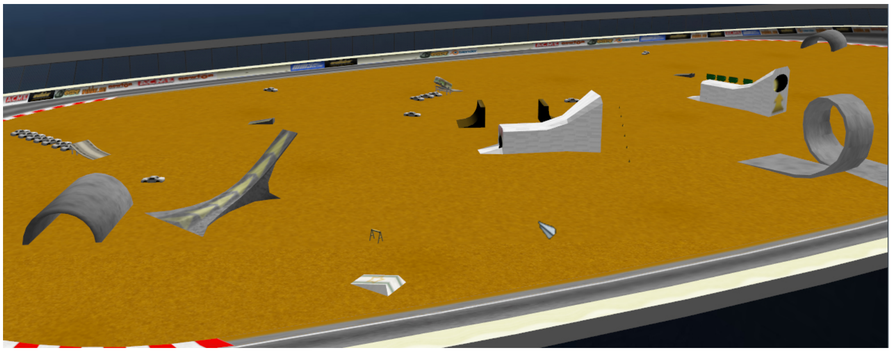


Figure 4.8: Stunt arena example, generated for a player modeled as low *Evel Knievel* and medium *Sunday Driver*

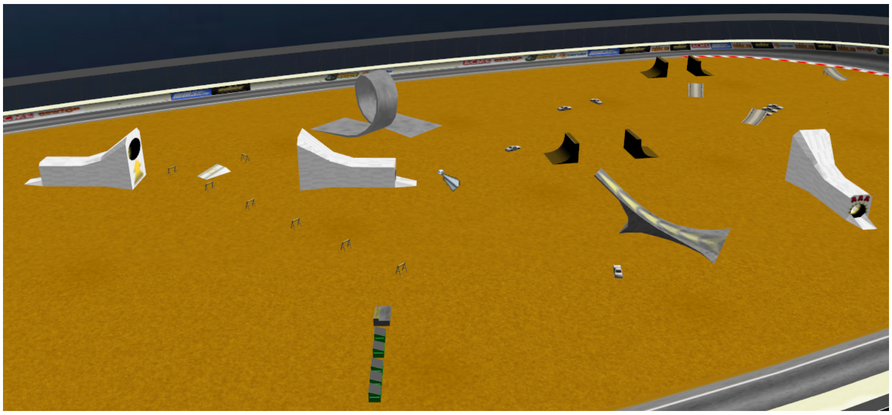


Figure 4.9: Another stunt arena example, generated for a similar input of a player modeled as in Fig. 4.8

Finally, we point out that these emergent cases did not occur as much as expected. We can identify two reasons for this observation. First, this can be explained by the nature of Stunt Playground, with a gameplay naturally biased towards high values of *Evel Knievel* models. The second reason relates to our player behavior modeling goals. In our case, emergence is only dependent on the dual simultaneous classification of a player as both *Sunday Driver* and *Evel Knievel*. We find two related reasons for this lack of emergence: (i) both measurements (see Section 4.3) are in fact less disjoint than we intended, to the point of actually acting as one single scale, and (ii) players simply do not behave like that.

4.5 Conclusions

In this chapter, we presented our results in applying gameplay semantics within a game, to generate player-matching game worlds. We demonstrated that our semantic-based methods are generic and re-usable enough to be integrated with an existing game, in this case Stunt Playground. This approach, based on designer-defined gameplay semantics, can use procedural content generation to create player-matching game worlds. These game worlds are not only fitted to the players behavior and experience, but also include the expected variability of procedurally generated content. We also confirmed that our generation process can have some emergent behavior, beyond what was specified by designers. Our semantic framework can therefore not only bridge procedural content generation, gameplay and designers, but also add some emergent, yet controllable, elements to the generation of player-matching game worlds.

5
Mobile adaptive procedural content generation

Procedural content generation can act as the adaptation mechanism in an adaptive game. One of the key research questions relates to which types of features can be generated in such a game. In this chapter we investigate the generation of level properties in response to a global experience-driven feature: difficulty. As a case study, we use a platform game in the mobile devices domain. The nature of most computer/console games is different from that of most modern mobile games, which are typically targeted at casual gamers and are played in a wide variety of space, time and device contexts. We argue that this feature of mobile games naturally fits with adaptive procedural content generation (PCG). In this chapter, we propose the integration of two PCG-based approaches (experience-driven and context-driven PCG) to support the generation of adaptive mobile game levels. We present and discuss the implementation of our approach in an existing game, *7's Wild Ride*. Similarly to Chapter 4, gameplay semantics and player modeling are used to steer a level generator, featuring a time-dependent dynamic difficulty adjustment mechanism. From our two user studies, we conclude that (i) context-driven levels have advantages over traditional ones, and (ii) the game can adapt to different player types, keeping its gameplay balanced.

Publication notice: *The scientific content of this chapter was published in the the fourth workshop on Procedural Content Generation in Games [Lopes 13b] and in the eighth International Conference on the Foundations of Digital Games [Lopes 13a]*

5.1 Introduction

The nature of most modern mobile games, *i.e.* games played in smartphones and tablets, is different from that of most computer/console games. Mobile gaming is becoming the preferred medium for the casual gamer, changing the demographics in the marketplace and bringing along cheaper, less powerful hardware [Farrell 12], which are used in a wide variety of environments and conditions. By observing the most downloaded games in mobile marketplaces like Apple's App Store and Google Play, we can confirm that a vast majority falls within the definition of a casual game [Nielsen Company 09]: "inexpensive to produce, straightforward in concept, easy to learn, and simple to play".

With their casual nature, these mobile games need to appeal to wider audiences. Accommodating player characteristics, styles or preferences for such a varied demographics is therefore a bigger concern. The usual solution for catering for casual players, offering shorter, simpler gameplay, tends to alienate some more demanding players, typically from the hardcore audience. Furthermore, mobile games are played in a wide variety of environments, conditions and time availability, dependent on the player context. Again, the usual approach to accommodate this diversity is to enhance the casual aspect of these games: keep them straightforward, simple and short. For this type of games, attracting and retaining all player types, while providing meaningful gameplay in a wide variety of contexts, raises additional challenges.

5.1.1 The case for mobile adaptive PCG

Adaptive games have the potential to be more personal, by adjusting their content or mechanics to better serve individual player needs [Lopes 11b, Yannakakis 11]. Ideally, adaptive games can accommodate for all types of players (casual or not) and their wide variety of commitment, skills or styles, *i.e.* player experience. Additionally, adaptive games have the potential for being meaningfully responsive to a variety of player-related contexts. For example, for a player waiting at an airport for a specific time period, an adaptive game could adjust the overall gameplay experience to fit that time-constrained context.

Currently, standard approaches for supporting such adaptive games are increasingly based on procedural content generation (PCG) [Lopes 11b, Yannakakis 11]. We argue that this adaptive PCG can tie in very naturally with the *casual* mobile gaming paradigm (henceforth, simply referred to as mobile games), where broad ranges of player commitment and circumstances should be addressed. To accommodate this wide variety of player experience and player-related contexts in mobile games, we envision two different modalities of adaptive PCG: experience-driven and context-driven.

5.1.2 Experience and context-driven PCG

Our research goal is to investigate the integration of experience and context-driven PCG in the mobile games domain, and to assess its potential value [1].

In *experience-driven* PCG, generative methods are responsive to some sort of player-generated *gameplay-specific* data. In this chapter, we propose a semantics-based experience-driven PCG approach where online content generation is used to dynamically adjust the game's difficulty. This dynamic adjustment of game difficulty (DDA) to the personal skill level of a player has the potential to attract and retain a larger variety of players.

We define *context-driven* PCG as the use of generative methods to yield content that fits some *player-related context*. The demands a given context puts on that content generation are therefore extrinsic to the gameplay, and respond to that player's concrete situation (e.g. available time, weather, location, health,...) Regardless of whether a context is explicitly input or implicitly derived, the neat effect will be a direct steering of the generator (even if players are unaware of it). Moreover, such demands of context-driven PCG can stretch deeper and more meaningful than players might anticipate. For example, for the airport time-dependent context mentioned above, setting a play duration constraint should not only determine the time to play, but also smoothly scale the full gameplay experience under that constraint. We envision that this form of adaptive PCG provides a powerful basis to accommodate for a variety of player-related contexts. In this research, we investigated a first example of context-driven PCG: level generation for an *explicitly* set play duration constraint.

Our case study in this chapter is a first step towards demonstrating that experience-driven and context-driven generation fit well together to support an adaptive mobile game experience. For this, we developed a game that allows you to specify how much available time you have to play; a level is then generated online, fitting both: (i) that time-constraint (context-driven PCG), and (ii) your measured skill level (experience-driven PCG). For this, gameplay semantics, *i.e.* gameplay information about the world and its objects, is used to support and steer the procedural content generator.

The above mentioned demands of mobile gaming (large variety of players, skills and contexts) cannot be met anymore by simply using handmade static levels. We believe that both experience-driven and context-driven game levels provide a much better and richer alternative, and that our adaptive PCG proposal is a valuable contribution to solve this mobile gaming challenge. Recent mobile games like *Canabalt* (2009) or *Robot Unicorn Attack* (2010) confirm this trend, by already incorporating a simple form of experience-driven PCG. Game content is there generated on the fly, based on player performance, but only sampled at the single instant generation is executed [Lager 09]. Our research, on the other hand, collects and uses player performance and time progression, over the whole game session, to steer generation.

[1] a video on this research can be found at http://rlop.es

This chapter is structured as follows: Section 5.2 outlines the methods behind our case study's generator. In Section 5.3, we explain our dynamic difficulty adjustment method and player modeling approach, including both our control mechanisms and the level features that can be generated. Section 5.4 presents and discusses our play testing results, preceding our conclusions in Section 5.5.

5.2 Content generator

To demonstrate our approach proposed in the previous section, we implemented an adaptive version of a mobile game developed by a multidisciplinary team for a course project at the Entertainment Technology Center of Carnegie Mellon University. The game, named *7's Wild Ride* (Fig. 5.1), is a side-scrolling platform game where players have to prevent the main character from falling off a rolling snowball. They keep the character in balance by: (i) tilting the mobile device left and right to counteract the unbalance gravity effects of navigating slopes, and (ii) by jumping on the snowball over obstacles that it picks up. Additionally, score points, power ups and achievements are part of the game. To balance the character, the player has to keep it within a safe zone, on the top of the snowball. If the player is riding the snowball in an unstable position, an animation is triggered to warn that the balance must be correct to avoid falling (see Fig. 5.1).

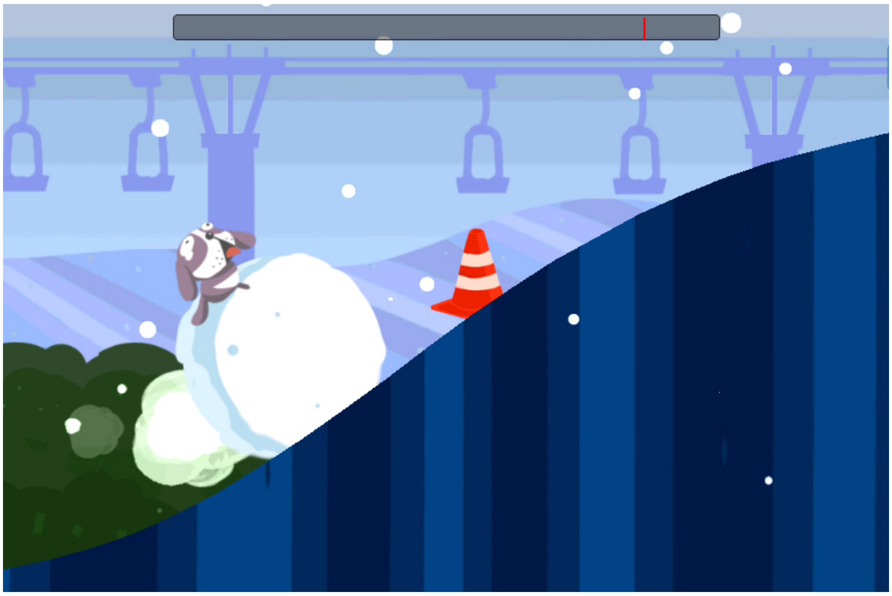


Figure 5.1: *7's Wild Ride*: player character is struggling with balance on the snowball due to an upwards slope (danger zone). Halfway the slope, the snowball is about to pick up the obstacle cone.

This mobile game was originally developed using a fixed set of designer-handmade levels, in the Unity game engine. Therefore, our first step was to develop a content generator able to create levels on-the-fly[2]. Subsequently, the control over the generator (as explained in Section 5.3) will support its dependency on player experience and context.

The generator creates levels by selecting and combining *level segments* (chunks) from a content library, as the player advances through the level. Level obstacles are also selected independently from the content library and placed on valid locations of the level segments. This means that level segments can be dynamically coupled with obstacles, to synthesize different combinations, thus ensuring high variability.

The generator is always running as the player advances through the level. When the player reaches the midpoint of a level segment, it immediately selects and retrieves the next chunk from the library, placing it accordingly. The character's speed and the hardware capabilities of current mobile devices ensure that online generation does not cause any performance drops. The framerate remains identical to the original game. Additionally, and since this game is a right side-scrolling game (going left is not allowed), previous level segments are removed, to save up memory. This means that, at any given moment, only a fixed number of level segments exist before and after the player's current position. Players are unaware of this due to the limited camera field of view (see Fig. 5.1). Fig. 5.2 shows examples of the level segments included in the library, and an example global view of a generated level, at a given instant.

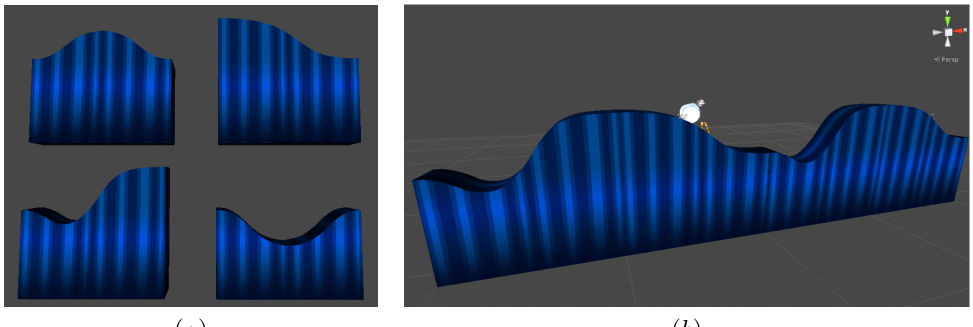


Figure 5.2: *(a)* Procedural level generation in *7's Wild Ride*: examples of chunk level segments, to be used by the content generator, *(b)* example of a level generated during play (debug view). Notice the two placed obstacles (next to the player and at the far right).

[2]from this point on, all *7's Wild Ride* mentions refer to the new adaptive version

5.3 Adaptive control

In this section, we describe the main methods used to support this on *7's Wild Ride* adaptive case study. Control over the content generator can be exerted by selecting which level segments, obstacles and obstacle locations to combine.

5.3.1 Semantics and DDA

To investigate experience-driven PCG in the context of our research goal, we implemented DDA on *7's Wild Ride*. We used virtual world semantics to control the procedural generator. In previous work [Tutenel 08], virtual world semantics is defined as all information about the world and its objects, beyond their geometry. This includes object properties, high-level attributes and functional information, as well as interrelationships (geometric, functional, etc) among different objects. We use the semantic library *Entika* [Kessing 12], to create a a hierarchical class (relational) database, responsible for storing all game objects and each of its associated semantics. As explained in Chapter 3, game designers use this library to specify semantics, atop game world geometry, in a generic and reusable way. Semantics imported from this library can be used as knowledge to automatically constrain and control PCG methods.

Gameplay semantics is all information that captures the gameplay value of all game world entities. It can express, for example, affective experiences, player performance, player actions or game actors. Importing gameplay semantics and embedding them in game worlds can be used to steer online PCG methods. This approach has already been successfully deployed to generate off-line experience-driven game worlds, which adapt to a player's behavior [Lopes 12].

For *7's Wild Ride*, we used gameplay semantics to support the DDA mechanism. Knowledge on the semantics of level segments and obstacle entities is specified, once and for all, prior to the game's deployment. The generator searches and selects content only if its semantics matches some desired input, typically expressed in the same semantic terms. An example is obstacle placement: locations for placement are only selected if they were marked as semantically valid.

Semantics was also used to label and select the difficulty associated with the different segments and obstacles in the library. Partnering with the designers team of the original *7's Wild Ride*, we were able to not only specify the new semantics, but also design the new DDA mechanism.

Three independent difficulty scales were defined: (i) slope selection, (ii) obstacle placement, and (iii) obstacle frequency. These were deemed sufficient to capture and influence the main difficulty-related aspects of the core gameplay. Semantics was pre-specified for all content, describing (i) the difficulty of the shape of a level segment (combinations of two level segments were also considered), and (ii) the specific difficulty of each valid location for the obstacle placement. For this, two independent numerical scales were defined.

Control over the generator is done by supplying input values for these three independent difficulty scales. For a certain input, the generator algorithm retrieves which library content was specified as appropriate to that difficulty scale. Additionally, obstacles placement follows a frequency constraint: only every N level segments is an obstacle selected and placed. Each input combination of the two difficulty scales might have several possible results (level segment and obstacle placement combinations). Whenever alternative results were deemed as equally difficult, the generator selects one randomly.

Gameplay semantics allows designers to participate in authoring the DDA mechanism without having to program. More importantly, it holds the expressive power to directly specify the building blocks and the selection rules which support level generation. Designers can create slope segments and even combine them to form groups with specific semantics. They can express constraints to the generation process, *e.g.* which segment can(not) follow which segment. Entities for obstacles can be created independently from the specification of valid level locations for their placement. And each of these entities and locations can be associated with numerical values for difficulty scales for obstacle placement or slope selection. The level generator uses all this information to synthesize (on-line) an adaptive level. With gameplay semantics, designers can even author DDA in an iterative way. Any corrections and changes to the semantic library directly tweak the level generator behavior. By changing the semantic attributes of placement and difficulty for all associated content, designers can effectively influence the behavior of the generation algorithm. This opens new possibilities for designers to explore the expressive power of this form of adaptive PCG.

To the best of our knowledge, this is the first time a semantics-based adaptive approach is applied in the mobile device segment. Android OS and the Unity game engine allowed quick integration with the technology supporting the persistent semantic library, SQLite and C#. We developed a light-version of a semantic engine, which is able to import and query semantic content from the database, as the game is running, on a mobile device.

5.3.2 Experience-driven PCG

The above semantics-based control of the generator already supplies a basis to support DDA. To fully realize this we developed a specific player model to steer the generator. Working together with the original team of *7's Wild Ride*, we became most familiar with its design goals and principles, and were therefore able to create this player model.

The proposed player model directly maps measured skills into the three difficulty scales described above. Player performance is measured and converted into a desired new level of difficulty for that player. Each difficulty scale is influenced by the following measured skills:

- slope selection: *BalancePerformance*, *BalanceDeath*
- obstacle placement: *BalancePerformance*, *ObstacleDeath*
- obstacle frequency: *ObstacleDeath*

BalanceDeath and *ObstacleDeath* flag whether the player died because of, respectively, loosing balance or hitting an obstacle. *BalancePerformance* measures the rate of transitions between the character's safe and danger zones (see Section 5.2). A player with less of these transitions is typically more skilled at balancing his character.

For all skills and difficulty scales, the same strategy was applied: an improvement on skill performance leads to an increment on the corresponding difficulty scales, and a decrease, to a corresponding decrement. Since all difficulty scales are independent, individual and specific behavior can be captured by the player model. This can lead to specialized reactions by the generator. For example, if the player is mostly dying from hitting obstacles, only the obstacle related difficulty scales are affected. Below is the algorithm for the mapping of skills performance into the difficulty scales they affect.

```
// All difficulty scales are pre-initialized to their minimum values

//transitions measured per time elapsed per level segment
//performance is normalized to the current difficulty
balancePerformance = transitions / (timeElapsed/obstaclePeriod);
balancePerformance = normalize(balancePerformance, slopeSelection);
//BalancePerformance
if(balancePerformance>balancePerformanceAvg OR transitions = 0)
        slopeSelection+=s1;
        obstaclePlacement+=o1;
else
        slopeSelection-=s2;
        obstaclePlacement-=o2;
//BalanceDeath
if(BalanceDeath AND timeAlive<timeAliveAvg)
        slopeSelection-=s3;
//ObstacleDeath
if(ObstacleDeath AND timeAlive<timeAliveAvg)
        obstaclePlacement-=o3;
        obstacleFrequency-=f1;
```

This player model is re-evaluated every N level segments, which might lead to difficulty scale changes. In our experiments for this chapter we used $N = 4$. If, in the current N segments, a player improves his *BalancePerformance* (compared to his past average) on the current N segments, the values for both the slope selection and the obstacle placement difficulty will increase. A decrease of these occurs in the opposite case. Both difficulty scales are affected, since jumping over obstacles can provide additional balance challenge. When the character dies and its last life duration was shorter than the average, difficulty scales are decreased accordingly, for the respective death type. The player model above only contemplates obstacle frequency decreases. The simultaneous presence of frequency increases and our context-driven PCG approach, explained in the next section, could lead to steep difficulty increases, overburdening players. All the increase and decrease operations should not lead to immediate difficulty changes. Therefore, we implemented two features to assure that the generator affects gameplay and difficulty only when continuous consistent performance improvements or declines occur. First, each increment is smaller than 1. Second, the generator only considers new inputs from the player model when a difficulty scale changes its integer value. Each of the difficulty scales are numeric scales: 1-5 for slope selection, 1-4 for obstacle placement, 0 to M for obstacle frequency. All the values for the parameters (N,M,s1,s2,s3,o1,o2,o3,f1) were determined experimentally in playtests with numerous players, as described in Section 5.4.

With this player model in place, player performance can be measured in terms of gameplay skills which are translated into dynamic difficulty scales. These scales steer the semantic generator to synthesize the appropriate content, classified according to that difficulty. This way, the difficulty of the level generated ahead is adjusted to match the player performance. In Fig. 5.3, you can see an example of the functionality of our experience-driven DDA, as observed in a user evaluation.

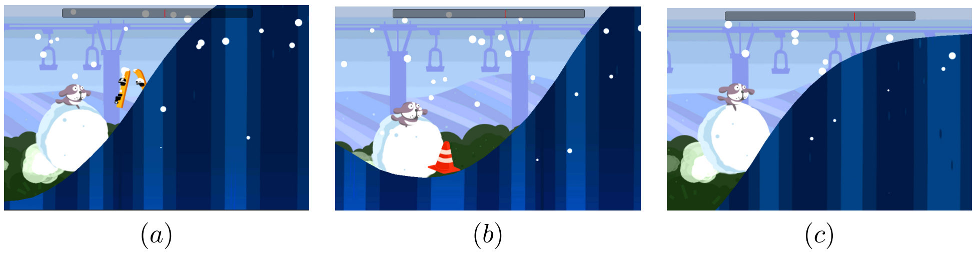


Figure 5.3: In *(a)*, player dies by not jumping successfully over the obstacle. Later on, in a subsequent life, an obstacle is placed at an easier location, in a similar situation *(b)*. Eventually, after repeated deaths by obstacle collisions, no obstacle is placed at all *(c)*.

5.3.3 Context-driven PCG

To investigate context-driven PCG, the generation of *7's Wild Ride* game levels has been made dependent on a time constraint (duration), explicitly specified by the player. The consequence of such a constraint is that it will directly steer the generator, adjusting the game to a context, in this case, the available time.

We implemented a solution to support this, that uses our generation approach, online recombination of level segments. A timer is responsible for ceasing level generation and gameplay, at the end of the requested duration. We opted for a strict strategy for this timer, where the pause and death menus do not interrupt time counting, even though no generation is occurring. In this way, we give high priority to fulfill the player's request, regardless of interruptions.

More importantly, and as explained in Section 5.1, setting a time constraint should also scale the full gameplay experience to that requested time. In our case, the requested time has a direct effect on the difficulty evolution of the game level, which is adjusted to fit not only the player experience, but also to fit the requested time.

This scaling relates to game progression. In many games, level difficulty typically increases with the advancement on a (handmade) game level, where the final sections provide harder challenges than the initial sections. The same happens between consecutive (handmade) levels, to provide the player a sense of progression.

In our case, we applied the same progression principle, but in the context of available time. As the player advances in the generated game level, approaching the requested time limit automatically increases difficulty to give the player a notion of proportionally scaled progression.

We start by dividing the requested time into five equal slots which are then grouped to create four time-based level phases: beginning (slot 1), middle (slots 2 and 3), pre-final (slot 4), final (slot 5). As you progress through these phases, all difficulty scales increase automatically by a value t, every P level segments. Both t and P change according to the phase the player is in. The closer a player approaches the final section, the more difficulty increases in frequency and value. As before, all parameter values were determined experimentally, with players, as described in Section 5.4.

With this approach, the context-driven PCG loop is closed: the requested time context affects difficulty increases, which has a direct effect on the generated content; this, in turn, scales gameplay progression to the available and requested time context. At the same time, this difficulty progression is still being balanced by the experience-driven DDA, which guarantees that any progression is still adjusted to each individual player performance. Ultimately, difficulty and content are being dynamically adjusted to both the individual player performance and the requested time context, thus creating a personalized and unique gameplay experience.

We are aware that PCG and, specifically, DDA can potentially prevent fair comparison of game scores and achievements, among players. We are interested in the debate on the (un)desirability of such comparisons, but that concern is not currently present in our research goals. Therefore, no scoring or collectibles game mechanics were taken into account in our case study.

5.3.4 Semantics and reusability

Our proposed approach can be generalized and applied to other games beyond *7's Wild Ride*. This is, in fact, a consequence of using a semantics-based generator, while saving considerable work.

The content generator can be applied to any game (most notably, platformers) which levels/world can be generated by the re-combination of adjacent world segments and the careful placement of adequate game objects. The semantic library combines all information about the individual content (geometry) with the knowledge that steers its retrieval and placement by the generator. In our case study, we defined the difficulty scales as these steering semantic attributes, which means they can be reused and extended in other projects. Using other generation control mechanisms beyond difficulty can also be easily achieved; it would entail using Entika to create and classify new semantic attributes, and minor changes to the retrieval process. As for our simple player model, it is certainly specific to *7's Wild Ride*, as it converts player performance data into our semantic difficulty scale values. Except for some generic machine-learning methods, typical player models are mostly dependent on specific games. However, with semantics, the implementation of a conversion from performance data to semantic attributes, as a new final step within such models, should assure their smooth integration with the generator.

As for our first experiment on context-driven PCG, this approach does not quite take advantage of gameplay semantics yet. In our future work, we plan to not only investigate new forms of context-driven PCG (beyond time), but also their integration with gameplay semantics. As with DDA, this would ensure their generalization and reusability.

5.4 Results and discussion

To evaluate our approach, we performed two user studies. We opted to use two distinct sets of participants to investigate each of our PCG-based methods: time context-driven and DDA. This not only avoids longer questionnaires, but also appeals to the different characteristics of both user groups, as explained below.

5.4.1 Time context-driven PCG

In our first user study, 17 participants (college students) played several game sessions and were interviewed. The goal was to define which parameter values (see Section 5.3) should be used in the generator, to guarantee a satisfying gameplay experience. Participants played two different versions of the game (different sets of parameters), and even re-played them by directly changing the parameters (available in these versions interface). This resulted in choosing the set of parameters to use in our second user study.

Since college students understand better the concept of limited time (when compared to our second user group), participants also evaluated our context-driven PCG approach. We performed a questionnaire to investigate the perceived value of our context-driven PCG implementation. Participants were invited to choose the time available for their game session, using a slider available in the game start menu. Additionally they were briefed on the meaning of our time-based level constraint, *i.e.* on the fact the game session would be generated to fit the requested time window.

We asked participants to rate, from 1(less) to 5(more), how valuable they thought this time context-driven generation was. Additionally, we asked them which type of levels they would rather play again, on their mobile devices: time-based generated levels or normal levels. 35% of participants ranked the value of time-based generation as 3, another 35% as 4 and 30% as 5. 65% of the participants would prefer to play time-based generated levels again, and 35% normal levels.

These results indicate the potential value in games that generate content in response to a player-specified time context. All the participants, even the minority which would not prefer this mechanism over traditional ones, recognized some positive value in this type of approach. It was clear from the interviews that everyone saw multiple applications for this method, even beyond time constraints.

Finally, we decided to ask about the role of adaptive PCG when it comes to comparing gameplay (scores, achievements). After explaining to participants to what extent adaptive PCG was supporting this game, 58% of the participants found the loss of gameplay comparison (between players) as important. This demonstrates that this is an important issue needing to be addressed. Finding ways of normalizing PCG-enabled gameplay and making dynamic game levels comparable (in terms of gameplay) remains an open question that deserves future research.

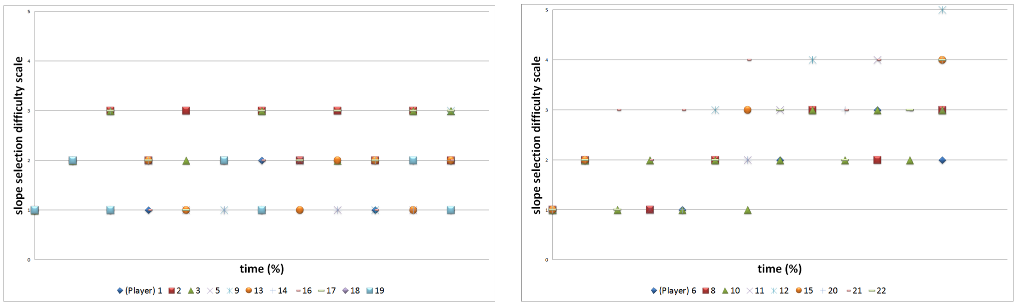
5.4.2 Dynamic difficulty adjustment

In our second user study, we focused on assessing our DDA case study. 22 participants (children between 6 and 12, visiting a science museum) played *7's Wild Ride*, using the set of parameters discovered in the first user study. Data about their performance was automatically logged and a short individual interview was conducted to assess the DDA mechanism. Before each game session (3 minutes), players had to complete a tutorial to learn the basic game mechanics: balancing and jumping. The children selected for this user study were identified *in loco* as rather casual players with little experience, thus with low commitment and skill.

Among other data, we logged the variation of the slope selection difficulty scale, measured throughout the game session for all participants. From the analysis of the collected data, we can identify two player categories: players without progression (Fig. 5.4a) and players with progression (Fig. 5.4b). For players without progression, the difficulty scale typically shifts a few units back and forth, between 1 and 3. For players with progression, the difficulty scale gradually increases more or less linearly, between 1 and 5. These patterns capture an underlying skill evolution, where players improve (or maintain) their performance. As expected, players without progression died more often (average of 6.72 deaths) than players with progression (average of 3.6 deaths), indicating this skill difference.

Challenge - We asked all participants to rate the challenge they felt throughout the game, from 1 (less) to 5 (more). Answers are summarized below, in Table 5.1.

Challenge results for rate 3 seem to indicate that the game is *balanced*, an indicator



(a) (b)

Figure 5.4: Variation of slope selection difficulty scale during the game session time, for all user study participants. Two categories were identified: *(a)* players without skill progression and *(b)* players with skill progression .

Table 5.1: Questionnaire ratings: game challenge and unfairness

	1 (less)	2	3	4	5 (more)
Challenge	0%	0%	55%	41%	4%
Unfairness	9%	45%	23%	15%	9%

of a successful DDA mechanism. However, the remaining participants (45%) felt a high degree of challenge. By correlating these answers with the two player categories from Fig. 5.4 (Table 5.2), we find some explanations.

Table 5.2: Questionnaire ratings: game challenge

Players	1 (less)	2	3	4	5 (more)
without progression	0%	0%	73%	27%	0%
with progression	0%	0%	33%	56%	11%

This seems to indicate that participants rated the experience as more challenging due to their improved skill progression and, consequently, to having reached higher difficulty scale values. Similar results on the obstacle placement difficulty scale confirm these observations.

Fairness - We asked all participants about the fairness of the game's progression. We wanted to detect whether the DDA mechanism was effective in providing the most appropriate balance, in a non-obtrusive way. Participants were asked to rate the unfairness felt throughout the game, from 1 (less) to 5 (more). Being children, this concept was hard to explain. Therefore, this question was posed with a more negative connotation (unfairness) than the challenge question, where participants would assess frustration (or satisfaction) and injustice. The answers are summarized in Table 5.1.

These results show that the majority of the participants found the game either fair and satisfying (rate 1,2) or balanced (rate 3). The positive/negative assessment nature of this question seems to further confirm our previous findings: again, the game seems *balanced*, an indicator of a successful DDA mechanism.

Further conclusions are observed if we correlate these answers with the two player categories from Fig. 5.4. As shown in Table 5.3, for players without progression, 64% rated the game fairness and frustration as 1,2 or 3 (fair, satisfying and balanced). For players with progression, 89% rated the same way. This seems to demonstrate that: (i) the majority of the participants is satisfied with the game balance of difficulty, and (ii) this satisfaction is stronger for players with progression. With these results, we observed that although gradual difficulty progression (Fig. 5.4b) implied a higher degree of challenge, that actually lead to a higher degree of satisfaction. This seems to indicate that the balance of difficulty to skill (DDA) was actually improved with the integration of a context-dependent difficulty progression.

Table 5.3: Questionnaire ratings: game unfairness

Players	1 (less)	2	3	4	5 (more)
without progression	9%	37%	18%	18%	18%
with progression	11%	56%	22%	11%	0%

Nevertheless, and since we had hoped for a balanced challenge level, we also made an effort to identify the reasons when that did not happen. We asked only the participants who identified high challenge (rate 4 or 5) to justify their answers. Although no one identified progression as the reason, 40% mentioned the new paradigm of using the accelerometer as the game controller, and 60% identified the "jumping over obstacles" mechanics. Logged data seems to confirm these answers: (i) players with jump issues died more performing jumps than everyone else (8.8 deaths to 7.6), and (ii) players with control issues died more of lack of balance than everyone else (6.25 deaths to 5.5).

Jumping seems to be a special case, deserving designers' attention. The concerns that it raised among players essentially relate to gameplay mechanics. In the game, the snowball picks up an obstacle (see Fig. 5.3b) which attaches itself to the snowball. To avoid the obstacle, the player has to jump on the snowball when the obstacle is approaching from behind, as it rolls on. Participants who identified the jumping as a challenge concern (and even others) often over-reacted by jumping as soon as they saw an obstacle on the screen, before it attaches to the snowball. They were mimicking behavior found in classic platform games, where the player character simply jumps above obstacles. These results give valuable feedback on the design of *7's Wild Ride* jumping mechanics, which might not be the most intuitive for some casual players.

5.5 Conclusions

In this chapter, we proposed the integration of experience-driven and context-driven PCG to support mobile adaptive game levels. We investigated this approach by implementing time-constrained level generation and DDA in the game *7's Wild Ride*, and evaluated it with two separate user studies.

Evaluation with users allowed us to conclude that our DDA mechanism can accommodate for a variety of players and skills. It has the potential to adjust the gameplay to different player categories (*e.g.* with and without progression), while keeping it balanced and players satisfied. User study participants were receptive on our context-driven PCG approach, to the point of valuing our time context-driven level generation above traditional mobile game levels, clearly recognizing its usefulness in the mobile games domain. Furthermore, our results allowed us to observe that the integration of context-driven PCG (specifically the gameplay progression it implied) with DDA actually leads to an increase in player satisfaction.

We can conclude that these two modalities of adaptive PCG can work well together in accommodating some of the characteristics of mobile gaming. The casual and context-dependent nature of mobile gaming seems to naturally call for this integration of PCG approaches, and we anticipate increasingly challenging research on this topic in the near future.

6 Authoring adaptivity in game world generation

So far, current research on adaptive games has not sought to actively include game designers in the creation loop. In our previous work, we focused on enabling the control over adaptive game world generation. In this chapter, we extend our contribution towards enabling designers to author adaptivity in game world generation, in a more expressive and specific fashion. We propose the use of *adaptation rules* to control adaptive game world generation. Adaptation rules are built atop gameplay semantics to steer the on-line generation of game content. Designers create these rules by matching sets of skill profiles, describing skill proficiency, with content descriptions, detailing the properties of game worlds. Game content is generated through the same matching and retrieval approach used in Chapter 3. We performed user studies with both designers and players, and concluded that adaptation rules can provide game designers with a rich expressive range to convey specific adaptive gameplay experiences to its players.

Publication notice: *The scientific content of this chapter was submitted for journal publication*

6.1 Introduction

Adaptive games are steadily becoming a focus of interest. As shown in Chapter 2, significant academic and even commercial work has been invested into games that dynamically adjust their content or mechanics to better fit individual player-dependent needs or goals.

However, current research on adaptive games is not concerned with actively empowering game designers to author adaptivity. Most work focuses on successfully establishing new methods for modeling the behavior of players or automatically generating game content or mechanics, in an adaptive fashion [Lopes 11b].

Such advancements already provide a solid basis to finally consider designers as part of an adaptive game creation loop. We are motivated to harness and integrate designers' specific knowledge into adaptive content generation methods. We believe such rich design knowledge still has a role to play, when authoring more dynamic games. Our goal is to contribute towards a design paradigm shift: from authoring *one* gameplay experience by using geometry models and static content towards designing *multiple ones* by using content generators. If appropriate and desired for the games in question, game creators will not design standardized game worlds but rather sets of instructions which will steer content generators, in-game, to dynamically create personalized worlds.

In this chapter, our main contribution is a semantic model to enable designers to author adaptivity in game world generation, in a more expressive and specialized fashion. We use gameplay semantics to support designer control over procedural content generation (PCG) methods and its integration with player modeling techniques. We build on top of our previous work to focus more on expressiveness and specificity of the control over adaptive game world generation.

We propose the use of *adaptation rules* to steer fully on-line adaptive game world generation. Designers specify these adaptation rules as matching sets of skill profiles and content descriptions. Skill profiles are created and visualized as radar charts, where each axis represents proficiency in a specific skill. Content descriptions allow specification of game world characteristics, expressed in terms of semantic game world entities. To support the use of adaptation rules, we implemented a new generation method where semantic game world entities are retrieved for a given input of a player model.

To demonstrate and evaluate our semantic model of adaptation rules, we implemented adaptive game world generation in a case study of an existing game. For this case study, we performed both design and play user tests, to assess whether this method can effectively be used to author adaptive game world generation.

This chapter is structured as follows. In Section 6.2, we outline our semantic model of adaptation rules and describe our generic generation algorithm. In Section 6.3, we present the case study and detail how it was integrated with our semantic model. In Sections 6.4, 6.5 and 6.6 we present and discuss the results of our user studies, before outlining our conclusions and future work in Section 6.7.

6.2 Authoring adaptive generation

In this chapter, as outlined in Section 6.1, we propose the use of adaptation rules to author and steer fully on-line adaptive game world generation. One of the key differences from our previous work resides in the authoring mechanism of such rules. We no longer rely solely on Entika and its text and drag-and-drop interface, as seen in Fig. 4.2 and Fig. 4.3. A more expressive, specific and intuitive method was implemented for this purpose.

6.2.1 Semantic model: adaptation rules

Adaptation rules are responsible for encoding the knowledge of game designers, by associating what they consider the most adequate game world characteristics to each given player profile. Rules act similarly as the cases in case-based reasoning, *i.e.* as the solution (the game world characteristics) to an observed problem (the player profile). Designers can construct an adaptation rule by creating an instance of a skill profile and associating it with a content description.

A skill profile groups a set of values for different players skills, where each one is a proficiency measured in a certain skill scale. An example of a player skill can be the ability to jump over platforms in Super Mario, measured by the percentage of successful jumps. Player skill profiles are best visualized and created in a radar chart, where each axis represents a specific skill. The shape of a skill profile is the polygon created by connecting all skill proficiency values in a radar chart, as observed in Fig. 6.1(a). If a certain instance of in-game player behavior, measured using these axes, occurs inside that shape, we say the player belongs to that skill profile.

A content description is a set of quantitative characteristics specifying how specific game world elements should look like. Examples include the gaps to be jumped on a Super Mario level (their number or size) or even constraints over them (their order or placement). A content description is also illustrated in Fig. 6.1(b).

As stated before, our new contributions focus on expressiveness and specificity of the control over adaptive game world generation. Specificity stems from the use of individual player skills (in skill profiles) and the open range of world entities that can be used in a content description (any object that can be included in a game world). Such range allows designers to create adaptation rules that can hold some highly specialized and therefore personalized value. For example, again in Super Mario, an adaptation rule can create more hardly accessible coins for a player who is excellent at collecting them.

Expressiveness links not only to this open range of entities but also to the lack of assumptions that our model holds. Our skill profiles make no assumptions on player behavior or psychology, and allow designers to freely decide, for example, where and when do beginners or experts stay on the scale. We also make no assumptions on the value or characteristics that the world entities in a content description might *mean*.

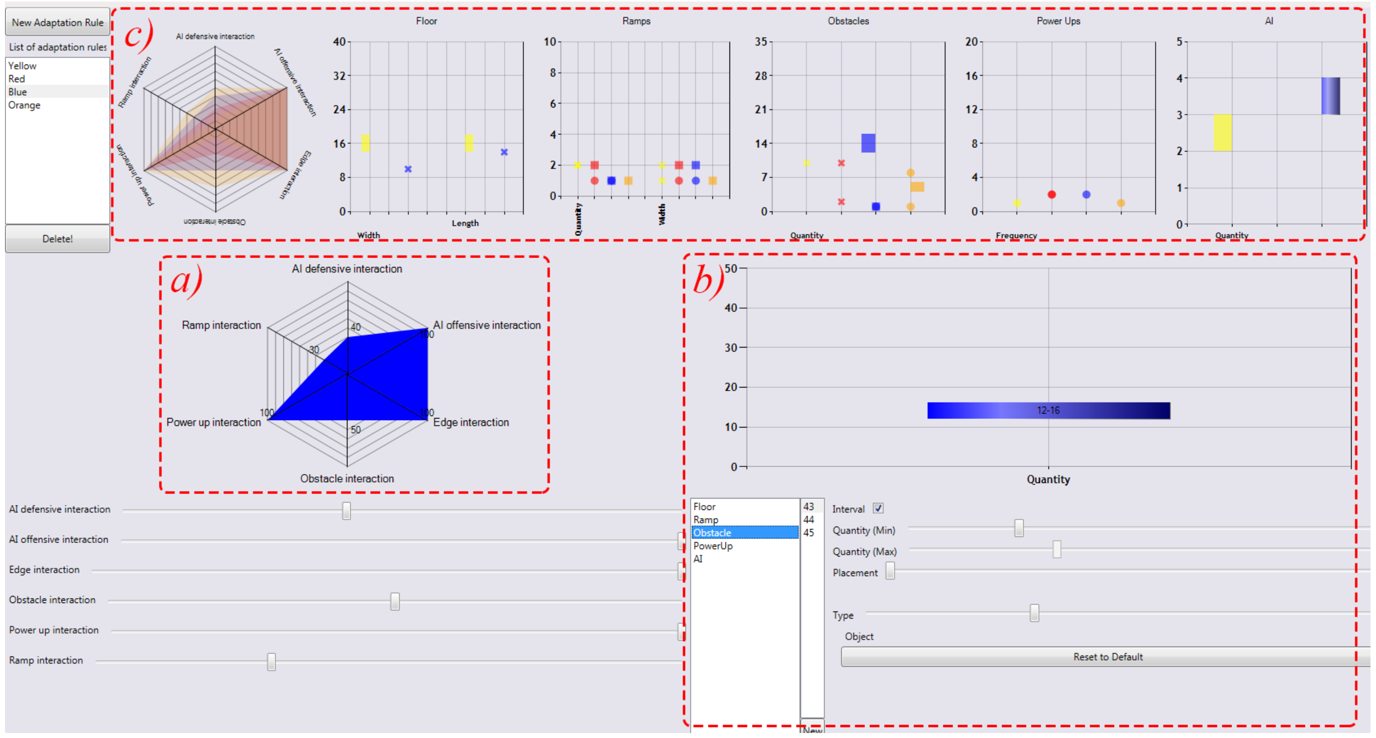


Figure 6.1: Design tool for creating adaptation rules. Skill profiles are created through the specification of their shape in a radar chart *(a)*. Content descriptions are associated to skill profiles and they are created through the instantiation of semantic entities and their relevant attributes. Content descriptions are visualized through point and range charts representing the attribute values for each semantic entity *(b)*. The tool includes an overall visualization of all adaptation rules, by showing an overlapping graph for skill profiles and a sequential graph for content descriptions *(c)*.

Content can be combined in any way to convey a desired experience, as intended by a designer.

Adaptation rules, skill profiles and content descriptions build on top of our previous work. Adaptation rules are represented internally as semantic gameplay descriptions, as proposed in Chapter 3. Skill profiles take the role of the preconditions of a semantic gameplay description, and content descriptions take the role of the semantics classes of a description. Unlike our previous work, semantic gameplay descriptions, here adaptation rules, are created directly by designers, and not automatically derived from classes labeled with gameplay semantics.

Skill profiles are represented internally by sets of player skill gameplay abstractions, as proposed in chapter 3. This data structure is already integrated within our overall gameplay semantics model and allows for parameterization of skills with proficiency values.

Content descriptions are represented internally by sets of semantic entities and corresponding semantic attributes. These classes are part of the semantic model proposed in [Tutenel 12] and already integrated with our gameplay semantics model, as outlined in Chapter 3. Like in our previous work and in [Tutenel 12], each semantic entity can have one or several geometric models that represent the geometry of that entity. These models are helpful for game world generation based on synthesis of game objects (*e.g.* as the layout solving in Chapter 4). As before, that is not necessarily the only case: semantic entities can include (or be included as input for) algorithmic procedures to generate the corresponding geometry. The research in this chapter includes the latter case. Such semantic entities still fit the definition of: all information about the game world and its objects, beyond their geometry.

Adaptation rules are created in a new design tool, expressively implemented for this research. Fig. 6.1 illustrates the tool. This design tool builds on top of Entika and it requires the previous creation of semantics for valid entities, attributes and player skills abstractions.

6.2.2 Rule matching and retrieval

With adaptation rules, the goal is (i) to identify when they apply in an adaptive game and (ii) to apply them to create an appropriate set of game content. Like in Chapter 4 and semantic gameplay descriptions, adaptation rules are the solutions for given adaptation problems, as outlined by designers. Identifying when rules apply, *i.e.* step (i) above, relies on the existence of a player model. As with our previous work, a player model should be responsible for capturing and classifying player behavior into a quantitative model. Significant changes in this model should reflect significant changes in player behavior, thus identifying the need for game adaptation.

A player model is also responsible for steering the selection of which adaptation rule(s) to apply, *i.e.* step (ii) above. This model, based on captured player behavior, should be matched with the skill profiles in adaptation rules to identify: (a) if the player is performing how a designer predicted and, (b) the content the designer specified as appropriate for that player behavior. Player models should therefore be expressed (or converted) in the same skill axes as the skill profiles in the adaptation rules. This allows an immediate matching process between the player model and adaptation rules.

In our previous work, as explained in Chapter 4, semantic gameplay descriptions are also matched against the player model and retrieved for use in content generation. In semantic gameplay descriptions, player preconditions are used to specify to what characteristics of a player a gameplay description applies. They are expressed as a player characteristic (skills, preferences, styles) and an upper limit value (*e.g.* jumping skill $< y$). We considered as a valid match all gameplay descriptions with player preconditions which are a *subset* of the input features of a player model. In other words, this meant that if a player is modeled as having proficiency value x in a certain skill, all gameplay descriptions which *include at least one* precondition on that same skill, and where $x < y$, are deemed as a valid match. The goal was to maximize variability and emergence by recombining several valid semantic gameplay descriptions.

For this research, we developed a new, and more powerful, matching method. The motivation behind it was to simplify the creation process for designers while being more certain about their intent, minimizing errors. When creating a rule, designers should be as precise as possible, and only have in mind a certain player profile. In Chapters 3 and 4, designers had to be aware that semantic gameplay descriptions might apply to players who only partially matched its player preconditions. Unlike before, we want to avoid forcing designers to be self-conscious that an adaptation rule might apply similarly. Although the previous matching method was not incorrect, it forced a more self-conscious design method. The following matching and retrieval algorithm for adaptation rules was created:

Algorithm 6.1: Matching and retrieval algorithm

```
FindRules(PlayerModel <$S_1, S_2, S_3, ..., S_N$>)
{
        for every AdaptationRule r
                Distance = CalculateEuclideanDistance(SkillProfile(r), PlayerModel);
                Axis = CalculateInclusion(SkillProfile(r), PlayerModel);

                if(Distance < MinD)
                        SelectedRule1 = r;
                        MinD = Distance;
                if(Axis <= MinA)
                        if(Distance < MinD2)
                                SelectedRule2 = r;
                                MinA = Axis;
                                MinD2 = Distance;

        if(SelectedRule1 == SelectedRule2)
                return SelectedRule1.Content;
        else
                return ContentFrom(SelectedRule1, SelectedRule2);
}
```

The input for matching and retrieving adaptation rules is a tuple $< S_1, S_2, S_3, ..., S_N >$ originated from a player model, where each item represents the proficiency value for a certain skill. For each input, we calculate the euclidean distance (formula below) between the input tuple and each tuple formed by the skill profile of each adaptation rules. We also calculate a measure of inclusion of the input tuple in the skill profile, by counting the number of axis where the corresponding value of the input is smaller or equal than the value of the skill profile (*CalculateInclusion*).

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}. \tag{6.1}$$

The initial idea behind this algorithm was to minimize distance and consider inclusion only, as stated before: if a tuple occurs inside the shape of a skill profile, the player belongs to that skill profile. However, we quickly decided on *not* making any assumptions on what the designers might think is best: inclusion on the shape of a skill profile vs. distance to that skill profile (even if the input behavior occurs "outside").

As such, our algorithm tries to find the adaptation rule with minimum distance (in terms of skill axis values) to the input player model tuple. Furthermore, it finds the adaptation rule with most axes with a value higher than the input tuple, and from those, at a minimum distance. Fig. 6.2 displays a case where two different rules are matched to an input tuple.

This method retrieves either one or two adaptation rules to apply at a given generation moment. If two adaptation rules are matched as valid, conflicting content descriptions might occur, *e.g.* generate 10 coins *vs.* generate 20 coins. In these cases, the method *ContentFrom* examines the content descriptions by observing if the same semantic entity (*e.g.* coins) occurs in both. If that is the case, one of the semantic entities is randomly removed. This means that the resulting final content is synthesized from merging the semantic entities of both rules (possibly with random removal). We feel confident that merging and random removal will result in meaningful content, as intended by designers, because they will occur mostly between two "close" adaptation rules. Furthermore, it is likely that a single adaptation rule will occur in a next generation moment. The resulting list of semantic entities, attributes and relationships represent a set of instructions and constraints (with possible geometric models attached) that will steer an in-game generator to create a game world, as exemplified in the next section.

The rule matching and retrieval algorithm (Algorithm 6.1) calculates *n* Distance and Axis values for each input player model value. This might raise some performance concerns, especially with on-line generation, since it does not scale with the number of adaptation rules. We believe this concern is not serious since we do not anticipate the number of adaptation rules to be high (>50). In the study case of this chapter, explained in the next section, we used on-line generation and a maximum

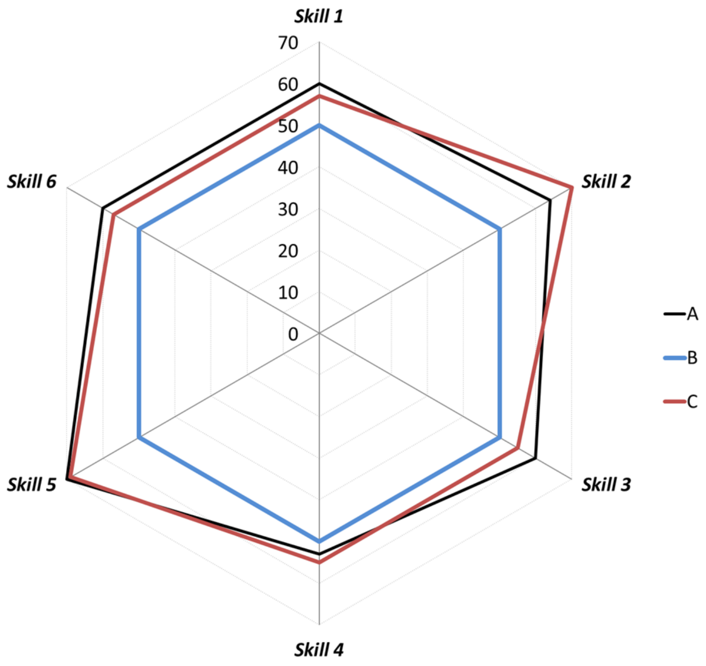


Figure 6.2: *A* represents an example input player model tuple, and *B* and *C* represent the two adaptation rules selected by the rule matching and retrieval algorithm. Rule *C* has the minimum distance to the input tuple. However, values for for Skill 2 and 4 are higher than the input tuple. Rule *B* has a higher distance to the input tuple, but all skill values are inclusive in the input tuple shape. To avoid assumptions on what is best, both rules are selected and combined.

of 50 adaptation rules with no performance issues. Furthermore, we anticipate that any future performance problems could be fixed with pre-computing (before game release) and indexing all distance values, for each adaptation rule, with all possible (typically discrete) player model values.

6.3 Case Study: Achtung Die Kurve 3D

To demonstrate and evaluate the contributions of this chapter, we integrated adaptation rules in an existing game, thus making it adaptive. For this case study, we experimented on a new technical domain, not yet tested in previous chapters: a 3D graphics-based game, with full on-line PCG, and as such, on-line adaptivity. Our aim was to demonstrate that, atop our previous contributions, gameplay semantics can also be applied to effectively support and control adaptive generation in this technical domain (3D and online).

6.3.1 Achtung Die Kurve 3D

Achtung Die Kurve 3D [1] is the game we chose to use as the case study for this chapter. As illustrated in Fig. 6.3, this is a third-person 3D version of the classic *Blockade* or *Achtung Die Kurve* games. In this game, players spawn at random places on a maze-like playing field composed of multiple floors connected by ramps. They keep growing ahead at a constant speed, using only left and right keys to control trajectory, until they crash. The goal is to survive and be the last one standing. Players leave a solid tail behind their moving head as they progress through a level. Crashes can occur when players collide with: their own tail, other players' tails, obstacles, floor edges, and ramp limits. As seen in Fig. 6.3, power ups can spawn at random places and be collected by the players. Power-ups are a combination of a target (self, only others, all) with a type (increase speed, decrease speed, turn harder, turn softer, switch keys, no tail, thicker tail, thiner tail and clear all tails, all with a temporary effect) and are visually identified using a color and icon system.

For this research, we implemented a modification of the original *Achtung Die Kurve 3D*. Our modification is a single player game, where the human player tries to reach the highest floor possible. Upon crash, the player tries again. Each new floor is generated automatically when the player enters a ramp and floors can be generated indefinitely. Opposing players are AIs who are constrained to the floor they spawn in, only trying to kill the human player.

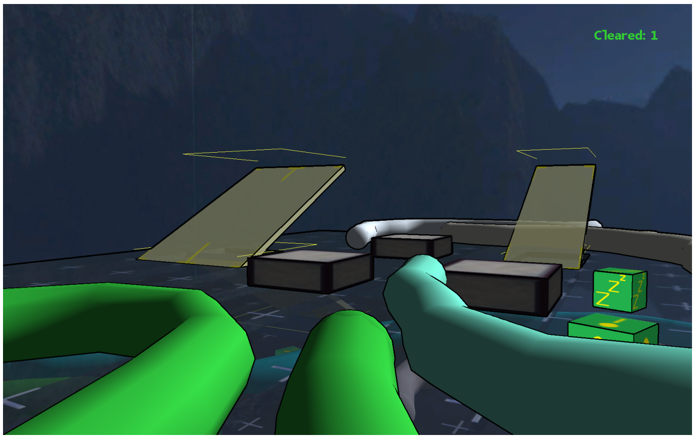
For our modification[2], we implemented the following changes in the original game: (i) we changed the scoring system from highest score on the number of surviving game sessions to highest score in cleared floors, (ii) we extended the AI behavior to avoid entering ramps, and (iii) we added the feature of different ramp widths (before, only a single fixed width was possible). Finally, our biggest changes were in the generation algorithm.

6.3.2 Floor generation

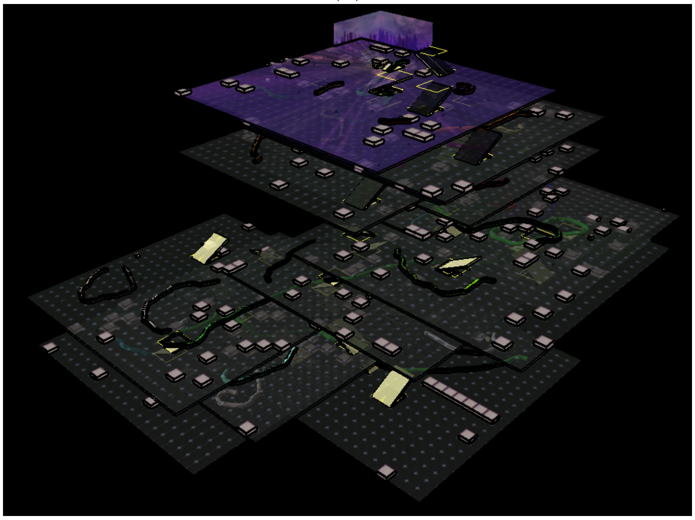
In the original *Achtung Die Kurve 3D* game, an off-line generation algorithm is responsible for creating a game world with 3 fixed-sized floors. This *ad-hoc* algorithm uses a three-dimensional matrix (n x m x k) to represent the game world, where each n x m matrix represents a floor. Each entry in this matrix represents a unit-size cell in the 3D environment. Matrix entries can represent: empty cells, obstacles, ramps, power-ups and AI spawn points. Microsoft XNA geometry primitives and shaders read this matrix and draw the game world accordingly. Before the game starts, the matrix is stochastically generated for a fixed requirement on the number of game world elements. At runtime, power-ups are stochastically generated in the matrix, according to a fixed time frequency.

[1]http://graphics.tudelft.nl/ mkt4/2011/groep7/

[2]A video of our modification of Achtung Die Kurve 3D can be observed in http://rlop.es



(a)



(b)

Figure 6.3: *Achtung Die Kurve 3D* game. In *(a)*, notice the player's green tail, on the left, the white and blue AI enemies, power ups on the right, and ramps and obstacles ahead of the player. In *(b)*, a tower of stacked floors was generated online, as the player progressed through the level

For our modification, we changed most of the matrix generation algorithm to support on-line PCG. We changed the inner representation of a three-dimensional matrix to a stacked list of matrices, with each new matrix being generated when a player enters a ramp. Each matrix (*i.e.* floor) can have n x m dimensions, different from the previous one. Each floor has only one entry, provided by the ramp used by the player to enter it. This entry is randomly placed in the generated floor. Ramp generation was extended to accommodate different widths, with the shader and collisions implementation changed accordingly.

Finally, we changed the stochastic nature of matrix generation to a more controlled method. Constraints have to be declared for each generated floor, typically describing number, type and placement of content. Sequential constraint solvers were implemented for the generation of: matrix (size), ramps, objects and AIs. In Sub-section 6.3.4 we discuss the format of these constraints in more detail.

6.3.3 Player model

As discussed in Chapter 3, an adaptive game requires a player modeling algorithm to capture the player's behavior. Furthermore, as discussed in Section 6.2, a player model is responsible for providing the matching input for retrieving adaptation rules. We designed and implemented a skill-based player modeling method for *Achtung Die Kurve 3D*.

Since our main focus is on authoring adaptive generation, we defined a clear set of design principles for our player model: (i) simplicity, and (ii) intuitiveness in matching to skill profiles. An investigation on more complex player modeling methods and its (comparative) effectiveness is beyond the scope of this research.

Our player model was inspired by the concept of experience points through practice, as observed in RPGs (*e.g.* Skyrim [Bethesda Game Studios 11]). The player model captures six individual players skills, on: (i) ramp use, (ii) obstacle avoidance, (iii) floor edge avoidance, (iv) power up selection, (v) AI defense, (vi) AI offense. We chose these six skills in order to capture all individual player behavior we could identify. Player behavior was identified by the observed in-game player actions (avoid, crash, use, kill), with relation to all content. Each skill is measured into an individual proficiency scale. The following algorithm illustrates how:

Algorithm 6.2: Player Model algorithm

```
PlayerModel(Event e, Target t)
{
        if(e == Avoidance)
                if(t == obstacle)
                        skillObstacleAvoidance++;
                if(t == floorEdge)
                        skillFloorEdgeAvoidance++;
                if(t == AI)
                        skillAIdefense++;

        if(e == Use)
                if(t == ramp)
                        skillRampUse++;
                if(t == PowerUp(help, self) || t == PowerUp(harm, others)
                        skillPowerUpSelection++;
                if(t == PowerUp(harm, self) || t == PowerUp(help, others)
                        skillPowerUpSelection--;
        if(e == Crash)
                if(t == ramp)
                        skillRampUse -= 1*ConsecutiveCrashes(t);
                if(t == obstacle)
                        skillObstacleAvoidance -= 1*ConsecutiveCrashes(t);
                if(t == floorEdge)
                        skillFloorEdgeAvoidance -= 1*ConsecutiveCrashes(t);;
                if(t == AI)
                        skillAIdefense -= 1*ConsecutiveCrashes(t);
                        skillAIoffense -= 1*ConsecutiveCrashes(t);

        if(e == Kill)
                if(t == AI)
                        skillAIoffense++;

        return <skillRampUse, skillObstacleAvoidance, skillFloorEdgeAvoidance,
             skillPowerUpSelection, skillAIdefense, skillAIoffense>

}
```

This player model increases or decreases an absolute proficiency value for each of the six skills. If a player is successful in avoiding a crash into a specific type of content, the corresponding skill is incremented. Successfully using ramps or killing AIs increases the corresponding skills. Power-up use detects if the used power up had a positive or negative effect for the player's success and increases or decreases the corresponding skill accordingly. Finally, crashing into a specific type of content decreases the corresponding skill (with no negative skill values though). However, that decrease is proportional to the number of consecutive crashes into the same content type.

For this player model, we implemented the detection of all these in-game events, which was not present in the original game. The occurrence of each event fires an update to the player model. Furthermore, each event is being persistently registered in a individual log file.

Our player model presents some noteworthy issues. Its reliance on absolute proficiency values (and not in, for example, percentages) is more effective and simple in capturing absolute practice proficiency. However, just as its inspiration, RPG experience points, it requires an elementary understanding of how its values are affected to grasp the meaning of an individual value. Furthermore, our player model makes the assumption that consecutive crashes (or deaths) of the same type strongly capture, in a linear fashion, a decrease in the corresponding skill.

6.3.4 Integration with adaptation rules

Integration with our semantic model of adaptation rules was required to make *Achtung Die Kurve 3D* adaptive. Three integration steps were needed: player model, generation parameters and the design tool for authoring adaptation rules.

The player model is responsible for invoking the algorithm for adaptation rules matching and retrieval (Section 6.2). In this case, the simplicity of the player model allows us to easily match it with adaptation rules, since they will use the same six skills in their skill profiles. This means the input (a player model tuple) directly matches with one or more skill profiles.

A matched adaptation rule will specify semantic entities and attributes in its content description. For *Achtung Die Kurve 3D*, we decided on the following semantic model:

- *Floor* entities have *Width* and *Length* attributes
- *Ramp* entities have *Quantity*, *Width* and *Placement* attributes
- *Obstacle* entities have *Quantity*, *Type* and *Placement* attributes
- *PowerUp* entities have *Frequency*, *Action*, *Type* and *Placement* attributes
- *AIs* have *Quantity* and *Placement* attributes

The non-numerical attributes require further explanation. The *Placement* attribute constrains the location to assign to that entity. It can hold three possible values: close to player, distant to player or random. The *Obstacle Type* attributes refers to specific *Achtung Die Kurve 3D* content, where obstacles can be an individual block (cells), a line, or stacked lines. As for *PowerUps*, *Action* refers to its positive or negative effect (help, harm or random) and *Type* to the target (self, others or random). As for numerical attributes, they can be expressed as either a fixed value or an interval, from which a random fixed value will be selected in-game.

This type of attributes determines that each adaptation rule can return, in its content description, several configurations of entities and attributes. This includes configurations using the same entity. For example, a content description might include 2 ramps of width 1 and 1 ramp of width 2. Furthermore, for this research we considered AIs as a specific piece of content, able to be generated. We felt this was a reasonable assumption since we are not changing (*i.e.* adapting) AI behavior but only its instantiation.

The floor generation algorithm, as described in sub-section 6.3.2, was implemented to accommodate the control constraints from the semantic model above. In other words, the input parameters of the generator are expressed in the same vocabulary and are able to steer it to create a floor with the specified characteristics. The placement constraint in the generator includes a hard-coded limit, proportional to the floor size, to determine between close or distant to player. As for power-ups, positive or negative effects are randomly mapped to specific ones: increase speed, decrease speed, turn harder, turn softer, switch keys, no tail, thicker tail, thiner tail.

An important aspect of this generator is its behavior in the absence of input parameters. In this case, the input parameters of the previous floor are re-used, on an *individual semantic entity* base. For example, if the input parameters are missing only the floor size, the previous values are used.

Finally, the design tool for creating adaptation rules was enriched with a semantic model that included the six matching player model skills and the semantic entities and attributes above. Using the tool's interface, see Fig. 6.1, designers can use this data to create adaptation rules by instantiating and shaping skill profiles and associating each of them with content descriptions. This semantic data is stored in a persistent database. This means that each instance of a database (*i.e.* a rule set) supports an instance of a different adaptive version of *Achtung Die Kurve 3D*. Upon start-up, the game loads all the rules from the database into memory and the matching and retrieval algorithm is ready to act.

6.4 Designing adaptive generation

In this chapter, we proposed adaptation rules, a generic semantic model that enables game designers to author adaptive game world generation, with a stronger expressive power and specificity than before. In the following sections, we assess its contribution by evaluating both the expressive and the specificity range of our authoring mechanism.

Our research question was to investigate whether game designers can control adaptive game world generation, in an expressive and specific way, to create a desired user experience, *i.e.* an adaptive gameplay experience. To evaluate our approach, we asked game designers to create a range of adaptive gameplay experiences in *Achtung Die Kurve 3D*. We then performed player testing in each of the created versions of the game, asking players about their user experience and logging their performance data. The goal was to discover if the players' experience matched with the designer's intent.

For the design experiment, our goal was to consider a wide range of different adaptive versions of *Achtung Die Kurve 3D* while still maintaining control and comparability within our experiments. For this, we selected 3 game designers, each of them tasked with creating the same set of 3 different adaptive versions of *Achtung Die Kurve 3D*. In the end of our experiment we had 9 different adaptive versions of the game.

The selected designers had amateur experience with game design and significant background in game technology, game development and content generation. They were all experienced gamers. The idea was to minimize the impact that professional pre-conceived design strategies might have in the experiment.

The 3 designers were asked to use our design tool to produce the following 3 variants, pertaining to 3 adaptive gameplay experiences:

- **Game 1**: Design a game that adapts to the player's skills, always maintaining a low challenge level;

- **Game 2**: Design a game that adapts to the player's skills, being easy to learn and hard to master;

- **Game 3**: Design a game that adapts to the player's skills in the most balanced and fair way;

Participants were instructed to design each gameplay experience for a typical 15 minutes game session. They were also advised to, faced with each task, be as specific or expressive as they wanted, by choosing to focus on the skills and content they deem as fit.

Each design session, comprising of the 3 variants, included: (i) demonstration of the game, (ii) explanation of all the concepts in this chapter, (iii) demonstration of the design tool, (iv) training with the game and the design tool, (v) iterative loop between designing adaptation rules and testing the resulting game, per task. Before (v), users were instructed to play a non-adaptive version of the game, for 15 minutes, to grasp and reflect on the evolution of the player model (always visible in this version). We observed all design sessions, providing support and engaging in open-dialog interviews.

6.4.1 Results

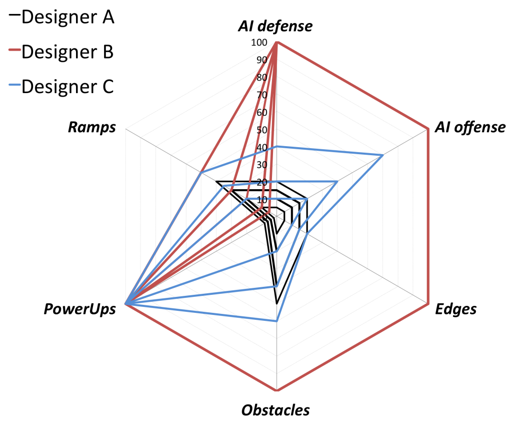
Fig. 6.4 (*a* to *c*) illustrates the skill profiles that all designers (*A*, *B* and *C*) created for for each game. They correspond to all adaptation rules created in theses experiments.

Designer A was fairly consistent throughout its games, by creating adaptation rules with similar skill profile shapes. This user focused on variation of all skills in a consistent manner, except for power up selection, which was ignored as a relevant skill. This designer increased the number of adaptation rules, using 4 in game 1, 6 in game 2 and 10 in game 3.

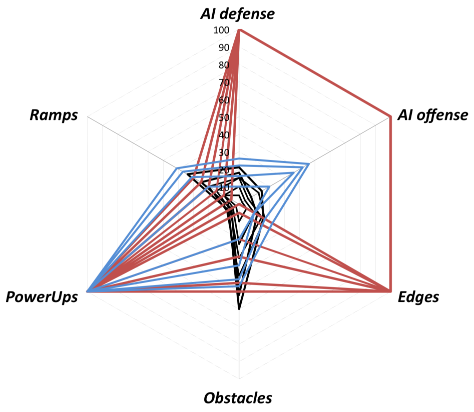
As for designer B, he varied the most between different games. In game 1 he chose to focus only on the ramp use skill, using 5 different adaptation rules. In game 2 he extended his focus on ramp use by also considering the obstacle selection skill, in 6 adaptation rules. In game 3, designer B focused on all skills except for floor edge avoidance and power up selection, using 11 adaptation rules.

Designer C was fairly consistent, using fairly similar shapes throughout its 3 games. For the first 2 games, and like designer A, this designer focused on all skills, except power-up selection. However, the shapes of the skill profiles are substantially different between both designers, since they are determined by the different used values in each skill axis. On game 3, designer C chose to only focus on ramp use, obstacle avoidance and AI defense.

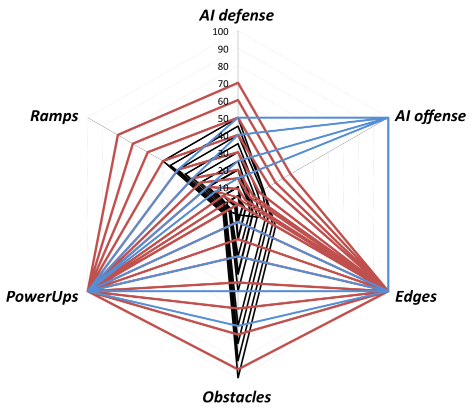
Figs. 6.5 to 6.7 illustrate the content descriptions that were used for each of these skill profiles. Each figure shows how the content, *i.e.* each semantic entity, corresponds to each skill profile. The attributes of each entity (*x axis*) correspond, from left to right, to the skill profiles of the corresponding Fig 6.4, from the inside to the outside. Regarding Floor and Ramps, each first attribute maps to its second corresponding attribute. So, for example, in Fig. 6.5, for the first adaptation rule (most inner rule for designer A, in Fig. 6.4) and for the Ramp entity, a configuration of 4 ramps of width



(*a*)



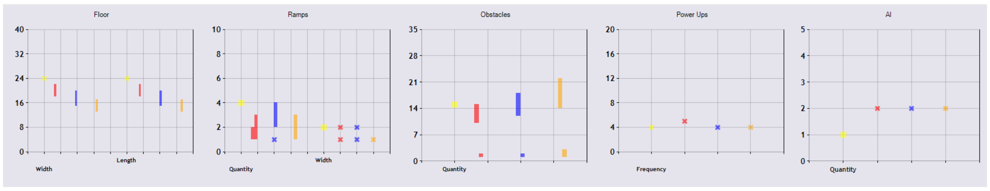
(*b*)



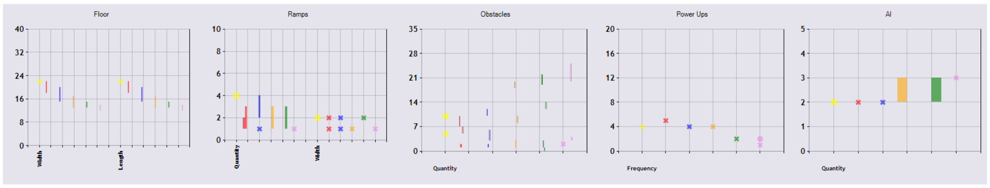
(*c*)

Figure 6.4: Skill profiles created for: (*a*) Game 1, (*b*) Game 2 and (*c*) Game 3, by all 3 designers (A, B, C)

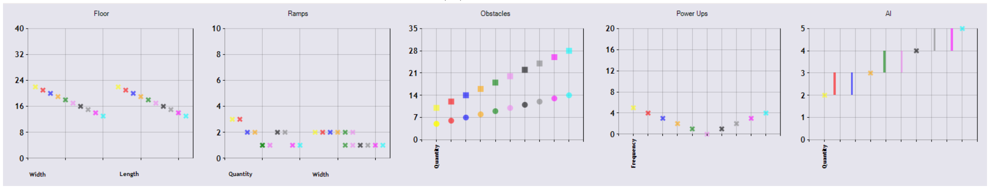
2 was declared. In these figures, a single point represents a fixed value, whereas an interval determines that a random value will be selected within that interval.



(a) Game 1



(b) Game 2

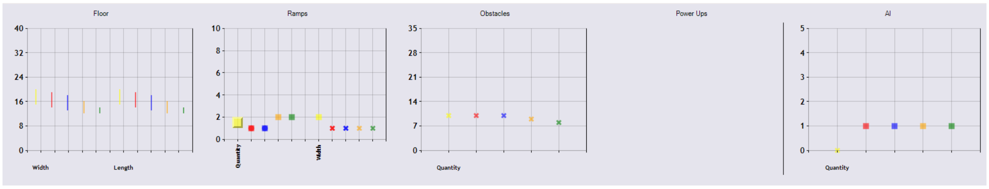


(c) Game 3

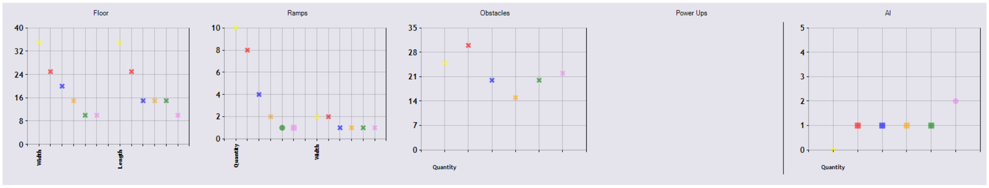
Figure 6.5: Content descriptions created by **Designer A**. Squares represent ”Distant to player” placement constraints and circles represent ”Close to player”. Crosses and intervals represent random placement.

Using these content descriptions, we can observe how designers created specific features for their adaptive games. For game 1, designer A chose to focus more on ramp and obstacle variation. Power ups and AIs vary little between adaptation rules. Size changes, but with a narrower noticeable variation. For game 2, besides slightly increasing the overall number of AIs, this designer maintains its attention to ramps and obstacles. In this game, when compared to game 1, the variation for both entities is wider. For example, for obstacles, rules for less skilled players include less obstacles than in game 1 and rules for more skilled players include more. Additionally, floor size variation has a more noticeable impact than for game 1. For game 3, designer A created more rules, and chose more gradual and wider variations on all types of content, including power ups (with a gradual decrease of helpful power ups, and increase of harmful ones).

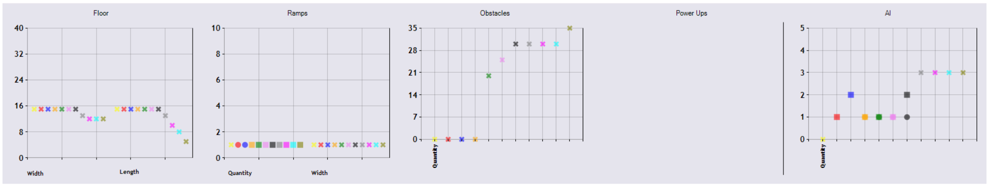
Designer B did not consider power ups in any of his games. For game 1, the designer used not only slight changes in how the content evolved, but also lower absolute values than the other designers. This is apparent in the absence of AIs in the first adaptation rule, and a variation of only the position of one AI in the following ones. For game 2, this designer chose to maintain a lower focus on AI variation, keeping it at low absolute values, and drastically change the way floor size and ramps varied. Obstacles still have an influence in this game, but they were made to be appropriate to the floor size in question, except for the adaptation rules for high-skilled players. In game 3, designer B created a very gradual adaptive experience. With more content descriptions, the designer used them to slowly insert slight variations at a time, typically changing only one aspect with each adaptation rule (*e.g.* increasing one AI at a time).



(*a*) Game 1. The emboss symbol in *Ramps* represents a "Distant to player" placement.



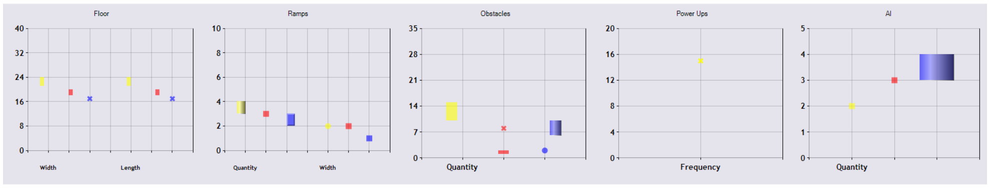
(*b*) Game 2



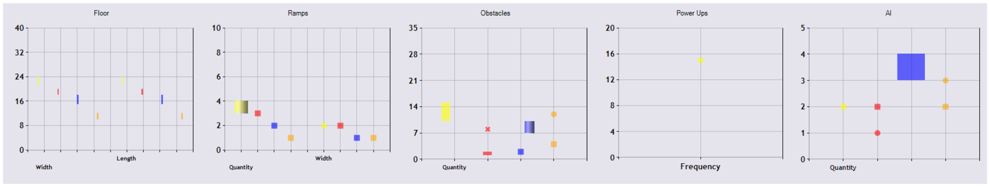
(*c*) Game 3

Figure 6.6: Content descriptions created by **Designer B**. The same symbols from Fig. 6.5 apply.

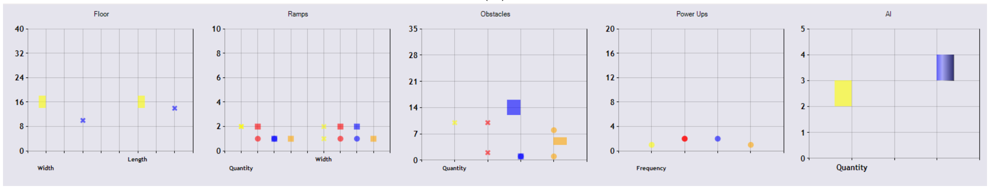
Designer C used less adaptation rules than any of the other designers. His games typically used a less wide variation in terms of content. In game 1, the designer changed the content very little, keeping it in a narrow variation interval. Even ramps can be more or less constant, due to the intervals used in ramp quantity in the first and third rule. The only noticeable specific aspect resides in a higher variation in AI quantity and position. For game 2, floor size and ramps changed the most drastically between rules. Obstacles and AIs also vary but using slightly overlapping attributes, when considering adjacent adaptation rules. For example, in AI quantity, we could observe an increment of 2, 3 and 3 in the lower skill profiles. For game 3, designer C focused more on gradual, specific changes on all attributes for ramps, obstacles and power ups. Floor size and AIs only change in the lowest and highest skill profiles.



(*a*)



(*b*)



(*c*)

Figure 6.7: Content descriptions created by **Designer C**. The same symbols from Figs. 6.5 and 6.6 apply. Additionally, intervals with a cylinder-like effect represent "Close to player" placement.

The following table summarizes what is most specific in each game, in terms of content descriptions, by detailing the semantic entities that change most frequently and with higher variation.

6.4.2 Discussion

The results of the design experiment (Figs. 6.4 to 6.7) allow us to report on the expressive range offered by our approach. Even asked for the same 3 variants, 3 different participants were able to design them in significantly different ways, resulting in 9 games with easily recognizable differences.

Skill profiles were able to offer flexibility at capturing the designers' desired intent for player characteristics. Designers A, B and C created significantly different shapes

Table 6.1: Content descriptions: specific focus in terms of semantic entities variation

	Game 1	Game 2	Game 3
Designer A	Obstacles, ramps	Floor size, obstacles, ramps	Everything (gradual)
Designer B	Everything (slight)	Floor size, ramps (drastic changes)	Everything (very gradual, one entity or attribute at a time)
Designer C	AI	Floor size, ramps	Ramps, obstacles and power ups

for their skill profiles. This was possible not only through a focus on different skill sets (*e.g.* notice the difference between designer A and B in game 2) but also due to the use of different values in each skill set. Furthermore, designers were able to experiment with different skill profile shapes between their own 3 different games (*e.g.* designer B and his 3 games). The concept of absolute values in the player model (as explained in section 6.3) was easily understood by designers. The selection of these values was made with certainty, with designers projecting the different desired player classes. However, in this type of experiment, the chosen values are influenced by the individual play testing (by designers) and by their assumptions about their ”target audience”.

As for content descriptions, they were able to offer an expressive range for designing 9 distinct games. As observed in Figs. 6.5 to 6.7 and Table 6.1, different designs naturally emerged from the variation offered by semantic entities and attributes, even within the same task. We argue that this is not only a result of the varied content, available to be combined, but mainly of the expressive power in the concept of adaptation rules. A good example is game 3, with: (i) designer A using all available content for very gradual but wide variations, (ii) designer B creating an adaptive experience where only one type of content varies at a time, very gradually, and (iii) designer C focusing on specific content (ramps, obstacles, power ups) on a narrower interval of variation.

Regarding specificity and the design experiment, it links with the available expressive range of our tool. As observed in Table 6.1, adaptation rules are a valuable tool to construct specific and recognizable features in an adaptive game. Designer B is the best example for this: in game 1 he chose to specifically focus only in the ramp use skill (Fig. 6.4*(a)*), and in game 3 he created a very particular content variation, already explained above. However, none of the designers chose to construct more complex specific adaptation rules, as we originally envisioned them. For example, combining several individual skills with individual semantic entities (*e.g.* ramp use skill variation resulting in ramp generation variation). The lack of this type of rules might be explained by the limited time of the (single session) design experiment and the heavier authoring burden of performing them.

Design sessions lasted for 3 to 4 hours, and we were able to gather several valuable observations and comments. Designers considered the design tool very expressive and were able to identify several ways of performing the assigned tasks, beyond their own designs. They were fully satisfied with their own designs. They spent most of the time exploring the tool's expressiveness and planning their designs. Typically, we observed that they chose to limit themselves in their designs to prevent dealing with an information overload for the time given. Furthermore, all designers enjoyed the intuitiveness of the radar chart polygons. One specific designer was very pleased with the lack of assumptions in our approach and its low level control (*i.e.* beyond, for example, pre-defined beginners vs. experts cases).

Overall, the design experiment allowed us to conclude that adaptation rules were able to provide an expressive and specific means to effectively capture the designer's intent in authoring adaptive game world generation.

6.5 Assessing adaptive gameplay

For the player experiment, our goal was to investigate if a desired adaptive gameplay experience could be conveyed through adaptation rules. The experiment consisted in play testing all the 9 previously designed variants of *Achtung Die Kurve 3D*, inquiring players about their experience and logging their performance data. To capture meaningful data, we considered 3 players per variant, thus performing tests with 27 players.

Each play session consisted of: (i) brief explanation of the game and its generative nature, (ii) brief training session with the game, lasting typically 2-3 minutes, (iii) play testing one of the games for 15 min. At the end of the session, players were requested to:

1. Plot how challenge/difficulty evolved over the time they played;

2. Answer the question: *Can you tell me specific level features you felt changed while you played and how?*

Players were asked to plot the perceived challenge over a grid representing the total game session time (*x axis*), with a minimum challenge of 0 and maximum challenge of 5 (*y axis*). Our aim was to match those plots with the desired adaptive gameplay experiences, as created by designers and see if, how and why they differ. Furthermore, we posed the second question above to confirm if the specific features of the adaptive game, as desired by designers, were easily felt by the players. We registered all the in-games events and player model values in individual log files, to correlate and support our user studies.

6.5.1 Results

Fig. 6.8*(a)* to 6.8*(i)* illustrate the results of the play test, for each player and each game. Each figure refers to a single game and includes the perceived challenge, as plotted by each of its 3 players. Below this, each figure also includes a time line plotting each instant where a crash occurred for each player. From all the logged performance data, we considered this to be the better measure to illustrate how challenge evolved over time and, as such, the better measure to correlate with each plotted line.

We registered all the players answers to question 2. Players were typically expressive in their answers, finding no difficulties in identifying level features. The table below summarizes these answers by focusing on which semantic entities were identified as the changing level features. We also asked *how* they changed, although this was a mere psychological mechanism to force reflection and therefore identify semantic entities. All the answers largely correlate with the plotted challenge lines, with players referring to increases and decreases of semantic entities.

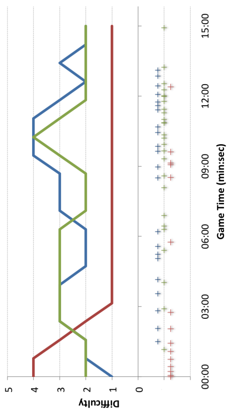
Table 6.2: Variation of specific level features, for each game and designer, as perceived by players. They typically refer to quantity or size of the semantic entity. If an attribute is between brackets, the player additionally identified it as a varying feature.

	Game 1			**Game 2**			**Game 3**		
A	ramps	obstacles	ramps, obstacles (position)	ramps, obstacles (both drastically), power ups	floor size, obstacles, power ups (type)	obstacles (positions), floor size, ramps, power ups (type)	ramps, AIs, obstacles	AIs, ramps, floor size, power ups (type)	floor size, AIs, obstacles
B	ramps (width), AI (position)	obstacles (position), AIs	AI (position)	floor size, ramps	ramps, obstacles (both drastically), floor size, AIs	floor size, obstacles	empty in beginning; ramps, obstacles, AIS (slowly)	ramps (position), AIs (slowly)	obstacles (position), AIs (smarter)
C	AIs (smarter), obstacles (type), ramp (width)	AIs (position), ramps (width)	AIs (smarter)	floor size, ramps, zero power ups	ramps, obstacles (both drastically)	ramps	ramps, power ups (type)	obstacles (position), ramps	ramps, power ups (type)

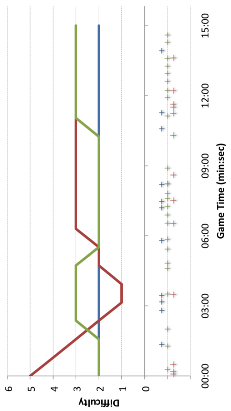
6.5.2 Discussion

The results of the player experiment (Figs. 6.8*(a)* to 6.8*(i)*) enable us to assess if the designers intended adaptive experiences were conveyed by adaptation rules and perceived by players as desired.

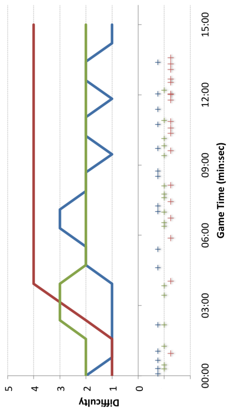
Overall, the challenge plots show that adaptivity is working and that the game is responsive to the players' performance. Typically, challenge does not remain constant throughout large periods of the game, but increases and decreases in alternate periods. The length and frequency of these periods and the range of the challenge varies per player and per game, but the plots show that the game offers a personalized dynamic challenge. This is visible even in game 2, a game with a clear tendency for gradually increasing challenge, where each increase step still includes some challenge variation.



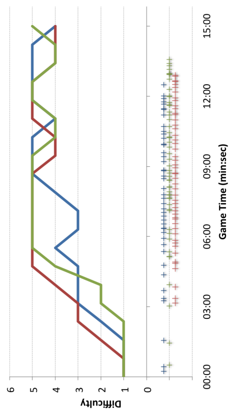
(*a*) **Game 1 by Designer A.** Players 1 (blue), 2 (red) and 3 (green)



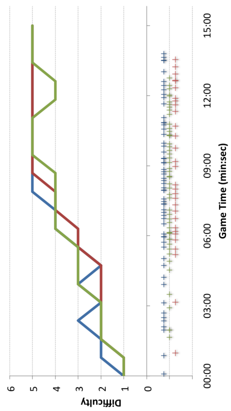
(*b*) **Game 1 by Designer B.** Players 4 (blue), 5 (red) and 6 (green)



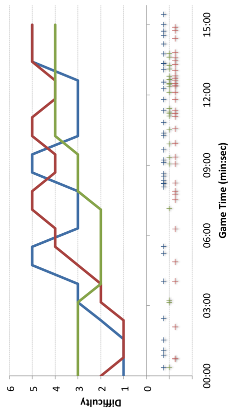
(*c*) **Game 1 by Designer C.** Players 7 (blue), 8 (red) and 9 (green)



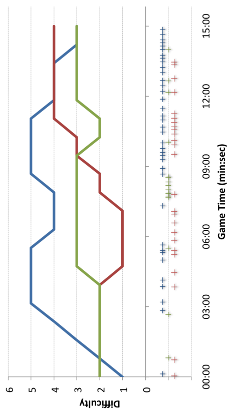
(*d*) **Game 2 by Designer A.** Players 10 (blue), 11 (red) and 12 (green)



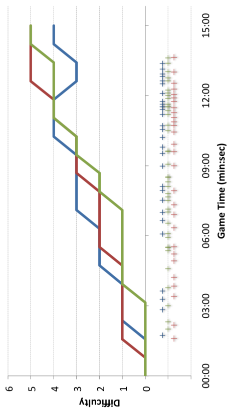
(*e*) **Game 2 by Designer B.** Players 13 (blue), 14 (red) and 15 (green)



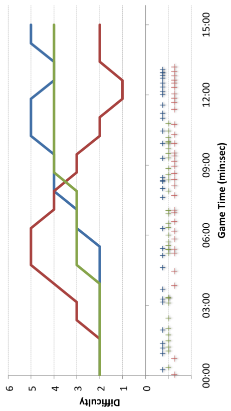
(*f*) **Game 2 by Designer C.** Players 16 (blue), 17 (red) and 18 (green)



(*g*) **Game 3 by Designer A.** Players 19 (blue), 20 (red) and 21 (green)



(*h*) **Game 3 by Designer B.** Players 22 (blue), 23 (red) and 24 (green)



(*i*) **Game 3 by Designer C.** Players 25 (blue), 26 (red) and 27 (green)

Figure 6.8: Difficulty/Challenge trend perceived by each player, per game, correlated with a timeline of the actual player crashing (death) events.

The exceptions for this behavior are player 2 and player 8. When observing player 2, it was noticeable that this version of game 1 (overall low challenge) eventually became too easy for the improving player, who became visibly bored with the experiment. We conclude that such boredom influenced his plot. Player 8 will be explained below.

These plots also show that the player perceived gameplay experiences match with the intent of the game designers. All versions of game 1 illustrate an adaptive game with a dynamic challenge variation, while keeping it at a low overall value. This is more evident if we compare the range of the challenge variation in all games 1 with games 2 and 3: most of the time is spent between challenge values 1 and 3.

In contrast, all versions of game 2 show a gradual increase in challenge, with most time spent in the higher values (between 3 and 5). This fits with the designers intent for game 2. An analysis of the created content (Figs. 6.5 to 6.7) shows that the task *"easy to learn, hard to master"* was interpreted by all designers as a game which is very easy for players with lower skills and extremely hard for players with higher skills.

Game 3 offered the most diverse results. This game was intended as the most possible balanced and fair game, and, because of that, able to best accommodate all types of players. With the exception of designer B (explained later), the versions of game 3 show a diverse challenge variation behavior (with values varying between 1 and 5) with no obvious tendency. Our conclusion is that these games *encourage* and accommodate a wide range of differently skilled players. For example, in game 3 for designer C, player 26 seems to quickly reach the top of his skills and is faced with higher challenges (subsequent failures result in a drop in challenge) while players 25 and 27 took more time to reach higher challenge levels.

Comparing these plots with logged data on crash time instants can confirm these insights. Our interest is in showing different *tendencies* in challenge variation, since it is not accurate to match an exact timeline (crash events) with a subjective one (user perceptions of challenge). The data on all versions of game 1 shows less crashes than any of the other games with a slight increase towards the middle of the experiment. Data on all versions of game 2 shows more crashes than any of the other games. They distribute with a lower concentration in the beginning of the game and a very high concentration towards the end. As for game 3, data is more diverse and does not show a clear tendency in the amount of crashes. However, its distribution seems to be more regular across all the timeline and, as with the remaining games, correlates with the perceived challenge, validating the observations in the previous paragraphs.

Some of the results require further analysis, since they are exceptions to these observations. In game 1 of designer A, both player 1 and player 3 peak (around the same time) at a challenge value of 4, a surprisingly high value for a version of game 1. This is easily explained by looking at the design data (Figs. 6.5 to 6.7). Game 1 of designer A is clearly the most difficult version of all games 1, with a much higher amount of obstacles, especially when correlated with the decreasing floor sizes. Furthermore, his skill profiles for this game are the smallest for all games 1 (Fig. 6.4), showing that it is easier for players to reach higher levels of challenge. Never the less, our observations are still valid, since designer A created his games 2 and 3 following on his game 1 design. As explained before for player 2, in this game we also found a complete mismatch between overall challenge and player natural skill. The same can be observed for game 1 of designer C and player 8. In this case, the player crashed very little in the beginning, playing for a lot of time without dying. During play test, we observed that when challenge actually increased, the player and his lack of skill could not cope with that slight but *sudden* change. Although we observed that challenge would also slightly decrease with his now worse performance, the significant psychological shock between the two phases of his play hindered his perception and influenced his ”binary” answer.

Another exception is game 3 of designer B. As explained before, this game features a very gradual adaptive experience, where slight variations are inserted at a time, typically changing only one aspect with each adaptation rule (*e.g.* increasing one AI at a time). The specificity of this game, as observed in Fig. 6.8*(h)*, allowed players to have a perfect consistent learning curve, with gradual challenge increase. Whereas the other games 3 show that they can accommodate different player types, this version shows a more balanced, linear form of challenge increase, especially when compared, for example with versions of game 2.

From these results we can conclude that the adaptation rules were effective in enabling individual players to experience the desired challenge fluctuations, either for maintaining a low challenge level, enabling a steep increase from low to high, or keeping a balanced fluctuation. The players’ answers and gameplay data typically matched with what was expected and what was designed for each game. We conclude that the adaptation rules’ expressive range can be effectively shaped by designers to author desired adaptive gameplay experiences.

Furthermore, from comparing Tables 6.1 and 6.2, we can observe that the specific features (variation in a type of content) of each game were easily felt and identified by players. All the players mentioned at least one of the specific features we identified in Table 6.1. For the players that mentioned more than these, the extra identified features were still correct. If we correlate those extra features with Figs. 6.5 to 6.7, we can observe that variation still occurs, albeit in a less drastic manner. Finally, by joining, per game, all player answers from Table 6.2, all specific features are covered.

6.6 Generalization

Encouraged by these results, we believe that the semantic model proposed in this chapter is highly generalizable. In this section we discuss the inherent features of our model that support such generalization (as well as its limits).

As mentioned before, adaptation rules make no assumptions on how skills are linked to content. For example, there is no obligatory number of rules or skill profiles nor is there a minimum or maximum amount of content that should be specified for each one. This lack of assumptions means that the authored associations between player behavior and game world content can be applied to any domain designers deem as valid. As such, this model is generic enough to be applied to any game genre where content generation can be considered an effective way of influencing gameplay.

Furthermore, the underlying building blocks supporting adaptation rules, *i.e.* player skill gameplay abstractions, semantic entities and semantic attributes, can all be created by designers to represent a certain domain of player skills and game world content (*e.g.* coin collection skill and coin entities). This means that adaptation rules can be as generic as the freedom to create such building blocks, *i.e.* as generic as its underlying semantic model [Tutenel 12]. The authoring effort to create these building blocks is outweighed by its potential for reusability. For example, power up entities can be used in many games, like *Achtung Die Kurve 3D* or *Super Mario*, its sequels or any game where power up generation is useful.

Skill profiles can be composed of any type and number of player skills. With no limitations on the number of the axes (or dimensions) of a skill profile, the model is generic enough to capture from the simplest to the most complex multi-dimensional behavior. However, skill profiles and underlying player skill gameplay abstractions are limited by: (i) its representation as a a scale of proficiency quantitative values and (ii) its matching to the skills measured by a player modeling algorithm to include in the game.

An additional limitation of our model resides on its dependency on skills. However, such limitation is weak and only exists since we decided to focus on skill-based games. This could be easily improved in the future since skills are only represented by a name and a proficiency value. Such representation could be used to capture player preferences, styles or other analogous features.

Content descriptions include combinations of semantic entities and attributes, which can represent a variety of types of content, *e.g.* any materials, parts, objects, spaces, scenes or worlds. As mentioned before, this means that content descriptions are as generic and powerful as its underlying semantic model and its authoring freedom, which were found to be considerably high [Tutenel 12]. Nevertheless, the potential for content descriptions is dependent on the existence of a specific generation algorithm, which can realize the semantic entities and attributes into a specific game world, *e.g.* the *Achtung Die Kurve 3D* floor generator.

A valid issue to raise is how our contributions relate to the burden of authoring adaptation rules and a compatible specific generation algorithm *vs.* the burden of authoring a single adaptive generation algorithm every time it is needed. Through its semantics nature, our approach enables reusability, where the semantic model and adaptation rules can be recycled in games with the same domain and (player) features. A typical example is the case of game sequels. More importantly, and as observed in our design experiments, adaptation rules enable an iterative loop between designing and testing, with no programming involved. Game designers can now test and experiment with controlling adaptive generation without the need to go back into the source code every time they need.

6.7 Conclusions and future work

In this chapter, we proposed the use of adaptation rules to author adaptive game world generation. Adaptation rules are built atop gameplay semantics to steer the online generation of game content. Designers create adaptation rules by matching sets of skill profiles, describing skill proficiency, with content descriptions, detailing the properties of game worlds. Personalized dynamic game worlds are generated through a matching and retrieval algorithm where designer-specified content descriptions are selected for a given input of a skill-based player model.

We integrated this approach in a specific game, *Achtung Die Kurve 3D*, and perform both design and play user tests. Through our design test results, we concluded that adaptation rules can provide game designers with a rich expressive range to control the adaptive generation of game worlds. This control over the dynamic and responsive generation of content can enable designers to author adaptive user experiences. Furthermore, this rich expressive range provides the freedom and the tools for game designers to be as specific in their intent as the existing skills and content allow them.

Through our play tests, we concluded that adaptation rules are effective in conveying the designed gameplay experience to players. We concluded that adaptation rules can provide a match between design and play, where the perceived and logged adaptive gameplay experiences correspond to the designers intent.

Through our case study and subsequent reflection on our method, we also identified its potential for generalization. Its lack of assumptions and the freedom in authoring a semantic model, both allow that this method can be easily adopted in a variety of game genres, as long as content generation can be used as an adaptation mechanism.

As for future work, we see room for various improvements. Regarding skill profiling in adaptation rules, we think that it would be interesting to add the possibility of weighing individual skills. The idea would be for the matching and retrieval process to differentiate between important from non-essential skills. This would allow a finer grained control over which skills are more important than others. Furthermore, we are interested in finding a more intuitive graphic interaction method for creating content descriptions, analogous to the skills' radar charts. Inspired by current research [Liapis 13], an idea would be to sketch a low resolution representation of the game world's entities and attributes.

Finally, we are interested in performing more and longer user tests with professional game designers. We believe their feedback will be instrumental in establishing whether our approach can be used to support a new design paradigm: the interactive authoring of adaptive game world generation.

7 Conclusions

In this final chapter, we conclude this thesis by discussing the present and future contributions that gameplay semantics can bring to the adaptive generation of game worlds. We start by revisiting the research contributions outlined throughout the chapters of this thesis, linking them to our original research questions. Then, we propose some recommendations for future work in semantics-driven adaptive games.

7.1 Research contributions

Although we have been observing a recent research interest in PCG-based adaptive games, we initially identified the lack of a corresponding growing trend in developing such games. We encountered a relevant problem in the absence of generic game technology to create adaptive games. No such technology exists, that it is accessible and controllable by game designers and powerful enough to be used across a variety of games and methods. We identified semantics as a valuable technique to be extended and used to reach that goal. Our initial definition of *gameplay semantics* allowed us to address such problems:

> gameplay semantics is the knowledge on the gameplay meaning and value of a game world and its objects.

Considering all this, we can now revisit our main research question:

> *How can gameplay semantics improve the adaptive generation of game worlds?*

To answer this question, we surveyed the field of adaptive games (Chapter 2) to identify research opportunities and existing problems. We confirmed that modern complex game worlds and their geometry are both a very effective way of influencing gameplay and among the most lacking *targets* of adaptive mechanisms. We concluded that the off-line and on-line generation of this type of game worlds, adjusted to the individual player, can become important *methods* for supporting adaptive gameplay. However, they still lack a declarative type of control that can be both linked to personal player behavior and exherted by game designers. To identify how gameplay semantics can empower these methods beyond such limitations, we addressed the following key questions, as introduced in Chapter 1.

1. How can gameplay semantics steer the adaptive generation of game worlds?

In Chapter 3, we addressed this issue by proposing a semantic generation framework for adaptive games. Through this framework, player and experience models, encapsulating personal behavioral data, are matched to gameplay semantics, as defined before. This semantics, holding gameplay meaning and value, is thus selected along with its matching game world content, specified as semantic entities and attributes. These act as the instructions, constraints or rules to generate a game world.

We used this approach throughout Chapters 4, 5 and 6. In Chapter 4, we modeled the player's style and matched it against appropriate game content (racing props), thus generating game worlds which increased fun for that player's style. In Chapter 5, we modeled player performance to steer the selection of matching game level sections and obstacles, synthesized to offer a dynamically adjusted difficulty. In Chapter 6, we captured individual player skills, in a multi-dimensional fashion, and generated maze-like floors from rule sets of matching game level characteristics. Gameplay semantics was responsible for steering game world generation from matching player styles, player performance and skills with the appropriate semantic content.

2. Which game world features can be generated from gameplay semantics?

In this thesis, we established gameplay semantics as the knowledge base able to steer different types of game world generation. In Chapter 4, we used layout solving, creating off-line generation algorithms able to synthesize game worlds through placement of (semantically player-matched) *objects*.

In Chapter 5, we developed an on-line generation algorithm based on the selection and re-combination of (semantically player-matched) *level segments* (*chunks*) and additional placement of *level objects*. In Chapter 6, we further continued with the on-line generation of game worlds, this time through a constraint solving approach, where playable *map-like grid levels* (representing mazes) were stochastically generated to fit (semantically player-matched) parameters. Furthermore, and outside the scope of this thesis, we collaborated with the integration of our gameplay semantics, in this case, player actions, to steer the generation of *connected spaces* through graph grammars [van der Linden 13a].

3. How can game designers use gameplay semantics to author an adaptive game world?

In Chapter 3, we laid out the principles on how gameplay semantics can be used to control adaptive game world generation. Game designers can specify gameplay knowledge semantic entities, *i.e.* game content. This effectively creates associations between specific content and specific gameplay, enabling generators to reason on such knowledge. Authoring such knowledge is therefore controlling how generation works.

In Chapter 4, we enabled designers to control the adaptive generation of racing arenas. They created gameplay semantic blocks associating player styles with gameplay experiences. These blocks were used to apply and tag different game objects or relationships between them. Authoring adaptivity consists in controlling which combinations of generated objects are appropriate for the possible player styles and experiences.

In Chapter 6, we aimed at making this process more interactive and expressive. We empowered game designers to create rules which associated player skills vectors (represented as radar chart shapes) with level content descriptions, expressed quantitatively in terms of semantic entities and their attributes. The creation paradigm was reversed from Chapter 4, to create gameplay semantic blocks (the skill vectors) and applying and tagging these with content. Authoring adaptivity consists in interactively creating such rules, which specify which game world configurations (the content descriptions) should be used for the possible player skills.

4. Which games, genres, player modeling and PCG methods can gameplay semantics apply to?

Throughout the chapters of this thesis, we experimented with several domains which allowed us to reflect on how and what can gameplay semantics apply to.

Regarding games and genres, we concluded that such scope lies in games which mechanics rely heavily on the geometry and objects of their world. In order to influence and adapt the player experience through game world generation, such experience should be strongly connected to the interaction with the surrounding world and objects. This excludes games which, for example, rely heavily on other elements, such as narrative interaction. Examples of suitable games can be racing games, with track and prop generation (Chapter 4), platform games, with level (terrain) and object generation (Chapter 5) or navigation-based games (*e.g.* maze-like games), with map and space generation (Chapter 6).

The integration of player models with gameplay semantics requires that they either are or can be converted into quantitative models. In this thesis, gameplay semantics used to characterize different player features (styles, preferences, skills, etc) use quantititative scales (fixed values or intervals). Therefore, the player models, responsible for matching themselves with such gameplay semantics, should be relatable to the same scales. Examples of such player models are heuristic-based tuple models (Chapters 4, 5 and 6).

We concluded that gameplay semantics is able to steer PCG methods that can be controlled through descriptions of game world objects or parameters. In semantics-driven adaptive game world generation, an input of player and experience models results in a list of semantic entities and attributes which are most appropriate, as declared by a designer. Generators must be able to use this list to create meaningful content. Typically, most constructive and search-based PCG methods can work in such fashion. For example, even evolutionary algorithms could include fitness functions which evaluate content on the basis of matching it to such a list of semantic entities and attributes. Other examples are generation through layout solving (Chapter 4), re-combination (Chapter 5) or stochastic constraint solving (Chapter 6).

To conclude, *gameplay semantics for adaptive game world generation*, proposed in this thesis, makes the following research contributions:

1. enables the adaptive generation of complex game worlds, by working as the *middleware* responsible for linking player models to specialized PCG methods;

2. opens up the possibility for generic adaptive generation, by being compatible and effective with a range of games, player model techniques and PCG methods;

3. enables and improves the authoring of adaptivity in games, by offering game designers an expressive and specific level of control over how player models can link to PCG methods;

4. improves and encourages the development of more adaptive games, by being an effective, generic and controllable method.

7.2 Recommendations for future work

We believe that our proposal of gameplay semantics provides a solid and rich basis for upcoming research on the fields of PCG and adaptivity. To conclude this thesis, we present some recommendations for future work in this direction:

Integration with event generation We believe that gameplay semantics could be used to control other types of generation, as scenario or narrative generation. It would be possible to associate gameplay semantics to other types of semantic entities beyond game world objects. Game events or player actions are an obvious example, since they can form the basis of scenario or narrative descriptions. Gameplay semantics could link player models to existing methods of scenario or narrative generation, thus enabling other forms of adaptivity beyond game world generation. More interestingly, by acting as a middleware, gameplay semantics could enable the integration of this type of adaptivity with this thesis approach to game world generation. Adaptive games would then include a coherent approach, where game events, and their associated content, would be generated in response to player behavior.

Automatic creation of gameplay semantics Our proposal of gameplay semantics defines it as the knowledge on the gameplay meaning and value of a game world and its objects. In this thesis, this knowledge is created by game designers, enabling them to author adaptivity. In alternative, an interesting future direction would be to investigate the emergence of gameplay semantics from actual player data. Gameplay semantics could be created from observing gameplay, by modeling both the player behavior and the content that enabled and supported that behavior. As such, the knowledge on the gameplay value and meaning of content would be a reflection of the mass collection of player data. Therefore, adaptive game world generation would be supported not by designers, but by past observed behavior from similar players. Furthermore, and more interestingly, such gameplay semantics could still provide a basis for a mixed approach, where game designers would use and fine tune player-created semantics to author adaptivity.

Adaptive game world interactions We believe that gameplay semantics could support other types of adaptive game mechanics, beyond generative-driven adaptivity. An interesting direction could be to make the interactions that a semantic game world offers dependent on the player behavior. This would imply that interactions would be adjustable, at run-time, to better fit the player. As simple examples, we imagine the amount of damage of a weapon (picked up by the player) as dependent on the player shooting skill, or the waiting period and/or speed of an elevator adjusted according to the players pacing style. All of these, and more, could be supported by gameplay semantics which would still link player models to semantic representations of game world entities. However, such representations would be used not to steer generation, but to choose and adjust parameters of available interactions, and how they can be performed.

Summary

When most commercial games are shipped, their gameplay has typically been prescripted. All game components are created during development, mostly as predetermined rigid artifacts with which a player will interact. This can lead to predictable and impersonal gameplay, while alienating unconventional players. Adaptivity in games has been recently proposed to overcome these shortcomings. Although adaptive games are a possible solution for this problem, in practice they are rare. The technology and development techniques to support such games are strongly *ad-hoc* and not easily accessible and controllable by game designers.

Procedural content generation and semantic modeling can powerfully combine to tackle these issues. This thesis proposes the use of *gameplay semantics*, *i.e.* the knowledge on the *gameplay* meaning and value of a game world and its objects, to adaptively generate game worlds.

Using gameplay semantics, we devised a generation framework aimed at creating personalized content for complex and immersive game worlds. This framework captures which content provided a given personal gameplay experience. This model is then used to generate content for the next predicted experience, through retrieval and recombination of semantic gameplay descriptions, *i.e.* case-specific mappings between content and gameplay semantics.

This framework can be used to link the procedural generation of game worlds with gameplay, as measured by player modeling techniques. Gameplay semantics is created in a generic way and can be effectively used to steer the procedural generation of player-matching game worlds, both at design and at game stage.

Gameplay semantics can steer the adaptive generation of game worlds by capturing the key features required for adaptivity. Both game world objects and properties can be synthesized in response to experience-driven features, *e.g.* the personalized difficulty of a game. Our player studies showed that this type of generation is successful in keeping such experience features in balance with different player types.

Gameplay semantics can be used to actively include game designers in the creation loop, by allowing them to author adaptivity in a more expressive and specific fashion. Designers can: (i) interactively create gameplay semantics that describe players and content, and (ii) match one with the other. These matching rules are the underlying semantic gameplay descriptions that support our generation framework. User evaluation shows that gameplay semantics can provide game designers with a rich expressive range to convey specific adaptive gameplay experiences to its players.

From the combined results of these contributions we can conclude that gameplay semantics is an effective, generic and controllable method to improve adaptation in games. We therefore hope that this work will encourage and facilitate the development of more and better adaptive games.

Samenvatting

De meeste commerciële games worden geleverd met voorgeprogrammeerde functionaliteit. Al de componenten waarmee de speler zal interacteren zijn vooraf ontwikkeld en weinig flexibel. Dit kan leiden tot voorspelbare en onpersoonlijke gameplay die vervreemdend is voor gevorderde spelers. Recentelijk is adaptiviteit in games voorgesteld om deze tekortkomingen te verhelpen. Echter in de praktijk komt dit nog maar weinig voor en zijn de ontwikkelingstechnieken om deze adaptiviteit te realiseren nog sterk ad-hoc en moeilijk toegankelijk en beheersbaar voor game designers.

De combinatie van procedurele model generatie en semantisch modelleren biedt mogelijkheden dit te verbeteren. De in dit proefschrift voorgestelde gameplay semantics koppelt de kennis over gameplay aan de inrichting van de game wereld en diens objecten, om zo game werelden adaptief te kunnen genereren.

Op basis van gameplay semantics, is een raamwerk ontwikkeld voor het ontwerpen van gepersonaliseerde content voor complexe game werelden. Dit raamwerk houdt bij welke eigenschappen geleid hebben tot een goede gameplay ervaring. Dit model kan dan worden gebruikt om nieuwe werelden te genereren door het opzoeken en hercombineren van geval-specifieke combinaties van wereldeigenschappen en gewenste gameplay semantics.

Dit raamwerk kan gebruikt worden om procedurele generatie van game werelden te koppelen aan gameplay, zoals vastgelegd door technieken die eigenschappen van spelers beschrijven. Gameplay semantics is gecreëerd op een generieke manier en kan effectief gebruikt worden voor het sturen van procedurele generatie van game werelden die tegemoet komen aan de gewenste gameplay, en dit zowel vooraf bij het ontwerp als tijdens de spelfase.

Gameplay semantics kan adaptieve generatie van game werelden sturen door eigenschappen van game werelden te linken aan de persoonsgebonden moeilijkheidsgraad van het spel. Onze experimenten met spelers hebben aangetoond dat dit soort generatie slaagt in het koppelen van dergelijke ervaringskenmerken aan uiteenlopende typesn van spelers.

Gameplay semantics biedt game ontwerpers de mogelijkheid om op een meer expressieve en specifieke manier de adaptiviteit te bepalen in het adaptieve creatie proces. Ontwerpers kunnen: (i) interactief gameplay semantics creeren die spelers en content beschrijven, en (ii) het n met het ander combineren. Deze combinatie

regels zijn de onderliggende semantische gameplay beschrijvingen die ons generatie raamwerk ondersteunen. Evaluatie van de gebruikers toont dat gameplay semantics designers een rijke, expressieve waaier van mogelijkheden biedt om specifieke gameplay ervaringen over te brengen aan de spelers.

Uit het geheel van de contributies in dit proefschrift kunnen we concluderen dat gameplay semantics een effectieve, generieke en controleerbare methode is om adaptiviteit in games te verbeteren. Daarom hopen we dat dit werk de ontwikkeling van meer en betere adaptieve games zal stimuleren en faciliteren.

Curriculum Vitae

Ricardo Lopes was born in June 27, 1982, in Lisbon, Portugal. In 2000, he finished his secondary school at Escola Secundária Nr. 2 da Portela de Sacavém, in Loures, Portugal. In 2008, he received his Master's degree in software and computer engineering from Instituto Superior Técnico, in Lisbon, Portugal. His thesis, called *Sketch-based interaction and calligraphic tags to create comics online*, introduced a calligraphic tagging interaction method to sketch comics in a web application. The main research focus was on automatic image retrieval using calligraphic tags.

After graduating, he worked as a software engineer for Exact Software, in the Netherlands. In the end of 2009, Ricardo started his PhD in the Computer Graphics and Visualization group of the Electrical Engineering, Mathematics and Computer Science Faculty of the Delft University of Technology. His research project was entitled *Gameplay semantics for the adaptive generation of game worlds*. In his research, that resulted in this thesis, he investigated semantic modeling techniques to support the development of generation-based adaptive games. As part of this project, he supervised several Bachelor's and Master's student projects.

Ricardo is currently working as a simulation software specialist at Macomi BV, in the Netherlands.

Bibliography

[Aldrich 02] Clark Aldrich. *A Field Guide to Educational Simulations*. Learning Circuits, 2002.

[Andrade 06] Gustavo Andrade, Geber Ramalho, Alex Sandro Gomes & Vincent Corruble. *Dynamic game balancing: an evaluation of user satisfaction*. In AAAI conference on Artificial Intelligence and Interactive Digital Entertainement, pages 3–8, 2006.

[Ashmore 07] Calvin Ashmore & Michael Nitsche. *The quest in a generated world*. In Proc. 2007 Digital Games Research Assoc. (DiGRA) Conference: Situated Play, pages 503–509, September 2007.

[Bakkes 09a] S. Bakkes, P. Spronck & J. van den Herik. *Opponent Modelling for Case-based Adaptive Game AI*. Entertainment Computing, vol. 1, no. 1, pages 27–37, 2009.

[Bakkes 09b] S. Bakkes, P. Spronck & J. van den Herik. *Rapid and Reliable Adaptation of Video Game AI*. IEEE Transactions on Computational Intelligence and AI in Games, vol. 1, no. 2, pages 93–104, 2009.

[Barber 07] H. Barber & D. Kudenko. *Dynamic Generation of Dilemma-based Interactive Narratives*. In Proceedings of the third Artificial Intelligence and Interactive Digital Entertainment conference (AIIDE '07), pages 2–7, 2007.

[Beal 02] Carole Beal, Joseph Beck, David Westbrook & Paul Cohen. *Intelligent Modeling of the User in Interactive Entertainment*. In Proceedings of the AAAI Spring Symposium on Artificial Intelligence and Interactive Entertainment, pages 8–12, 2002.

[Bethesda Game Studios 11] Bethesda Game Studios. *The Elder Scrolls V: Skyrim*, 2011. Bethesda Softworks.

[Bidarra 10] Rafael Bidarra, Klaas Jan de Kraker, Ruben M. Smelik & Tim Tutenel. *Integrating Semantics and Procedural Generation: Key Enabling Factors for Declarative Modeling of Virtual Worlds*. In Proceedings of the FOCUS K3D Conference on Semantic 3D Media and Content, Sophia Antipolis - Méditerranée, France, February 2010.

[Bielikova 08] M. Bielikova, M. Diveky, P. Jurnecka, R. Kajan & L. Omelina. *Automatic generation of adaptive, educational and multimedia computer games*. Signal, Image and Video Processing, vol. 2, no. 4, pages 371–384, December 2008.

[Blackman 05] Sue Blackman. *Serious games...and less!* ACM SIGGRAPH Computer Graphics, vol. 39, no. 1, pages 12–16, 2005.

[Braben, D. and Bell, I. 84] Braben, D. and Bell, I. *Elite*, 1984.

[Brathwaite 09] Brenda Brathwaite & Ian Schreiber. Challenges for Game Designers. Charles River Media, Inc, 2009.

[Chanel 08] Guillaume Chanel, Cyril Rebetez, Mireille Bétrancourt & Thierry Pun. *Boredom, engagement and anxiety as indicators for adaptation to difficulty in games*. In Proceedings of the 12th International Conference on Entertainment and Media in the Ubiquitous Era, pages 13–17, New York, NY, USA, 2008. ACM.

[Charles 05] D. Charles, A. Kerr, M. McNeill, M. McAlister, M. Black, J. Kucklich, A. Moore & K. Stringer. *Player-Centred Game Design: Player Modelling and Adaptive Digital Games*. In Proceedings of DiGRA 2005 Conference: Changing Views - Worlds in Play, pages 285–298, 2005.

[Chen 05] S. Chen & D. Michael. *Proof of learning: assessment in serious games*, 2005.

[Chen 08] Guoning Chen, Gregory Esch, Peter Wonka, Pascal Müller & Eugene Zhang. *Interactive Procedural Street Modeling*. In SIGGRAPH '08: Proceedings of the 35^{th} Annual Conference on Computer Graphics and Interactive Techniques, vol. 27, pages 1–10, New York, NY, USA, 2008. ACM.

[CNN 11] CNN. *Why most people don't finish video games*. http://edition.cnn.com/2011/TECH/gaming.gadgets/08/17/finishing.videogames.snow/(as of October 2013), 2011.

[Compton 06] Kate Compton & Michael Mateas. *Procedural Level Design for Platform Games*. In Proceedings of the Second Artificial Intelligence and Interactive Digital Entertainment Conference, pages 109–111, 2006.

[Core 06] Mark G. Core, H. Chad Lane, Michael van Lent, Dave Gomboc, Steve Solomon & Milton Rosenberg. *Building explainable artificial intelligence systems*. In Proceedings of the 18th conference on Innovative applications of artificial intelligence - Volume 2, pages 1766–1773, 2006.

[de Carpentier 09] Giliam J.P. de Carpentier & Rafael Bidarra. *Interactive GPU-based Procedural Heightfield Brushes*. In FDG '09: Proceedings of the 4^{th} International Conference on the Foundations of Digital Games, Florida, USA, April 2009.

[De Troyer 09] Olga De Troyer, Wesley Bille & Frederic Kleinermann. *Defining the Semantics of Conceptual Modeling Concepts for 3D Complex Objects in Virtual Reality*. In Journal on Data Semantics XIV, volume 5880 of *Lecture Notes in Computer Science*, pages 1–36. Springer Berlin / Heidelberg, 2009.

[Deussen 98] Oliver Deussen, Pat Hanrahan, Bernd Lintermann, Radomír Měch, Matt Pharr & Przemyslaw Prusinkiewicz. *Realistic Modeling and Rendering of Plant Ecosystems*. In SIGGRAPH '98: Proceedings of the 25^{th} Annual Conference on Computer Graphics and Interactive Techniques, pages 275–286, New York, NY, USA, July 1998. ACM.

[Doran 10] Jon Doran & Ian Parberry. *Controlled Procedural Terrain Generation Using Software Agents*. IEEE Transactions on Computational Intelligence and AI in Games, vol. 2, no. 2, pages 111–119, June 2010.

[Dormans 10] Joris Dormans. *Adventures in level design: generating missions and spaces for action adventure games*. In Proceedings of the 2010 Workshop on Procedural Content Generation in Games, pages 1:1–1:8. ACM, 2010.

[ESA 13] ESA. *2013 Essential Facts About the Computer and Video Game Industry*. http://www.theesa.com/facts/pdfs/ESA_EF_2013.pdf(as of October 2013), 2013.

[Etheredge 13] Marlon Etheredge, Ricardo Lopes & Rafael Bidarra. *A generic method for classification of player behavior*. In Proceedings of the the second workshop on Artificial Intelligence in the Game Design Process., 2013.

[Fairclough 06] C. Fairclough. *Story Games and the OPIATE System*. PhD thesis, University of Dublin - Trinity College, Department of Computer Science, 2006.

[Farrell 12] Michael B. Farrell. *Apps shake up video game industry*. The Boston Globe, November 2012.

[Gain 09] James Gain, Patrick Marais & Wolfgang Strasser. *Terrain Sketching*. In I3D '09: Proceedings of the Symposium on Interactive 3D Graphics and Games, pages 31–38, New York, NY, USA, 2009. ACM.

[Game Front 12] Game Front. *Why Dont Gamers Finish Games?* http://www.gamefront.com/why-dont-gamers-finish-games/(as of October 2013), 2012.

[Gilleade 04] Kiel M Gilleade & Alan Dix. *Using frustration in the design of adaptive videogames*. In ACE '04: Proceedings of the 2004 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology, pages 228–232. ACM, 2004.

[Gomboc 05] Dave Gomboc, Steve Solomon, Mark G. Core, H. Chad Lane & Michael van Lent. *Design recommendations to support automated explanation and tutoring*. In Proceedings of the Fourteenth Conference on Behavior Representation in Modeling and Simulation, 2005.

[Harteveld 10] Casper Harteveld, Rui Guimarães, Igor Mayer & Rafael Bidarra. *Balancing Play, Meaning and Reality: The Design Philosophy of LEVEE PATROLLER*. Simulation and Gaming, vol. 41, no. 3, pages 316–340, 2010.

[Hartley 09] Thomas Hartley & Quasim Mehdi. *Online action adaptation in interactive computer games*. Computers in Entertainment, vol. 7, no. 2, pages 28:1–28:31, 2009.

[Hastings 09] E. Hastings, R. Guha & K. Stanley. *Automatic content generation in the Galactic Arms Race video game*. IEEE Transactions on Computational Intelligence and AI in Games, vol. 1, no. 4, pages 245–263, 2009.

[Houlette 04] Ryan Houlette. *Player Modeling for Adaptive Games*. In AI Game Programming Wisdom II, pages 557–566. Charles River Media, Inc, 2004.

[Hullett 09] Kenneth Hullett & Michael Mateas. *Scenario generation for emergency rescue training games*. In FDG '09: Proceedings of the 4th International Conference on Foundations of Digital Games, pages 99–106, New York, NY, USA, 2009. ACM.

[Hunicke 05] Robin Hunicke. *The case for dynamic difficulty adjustment in games*. In Proceedings of the 2005 ACM SIGCHI International Conference on Advances in Computer Entertainment technology, pages 429–433. ACM, 2005.

[IGN US 11] IGN US. *GDC: 72 Percent of Players Finished Heavy Rain*. http://uk.ps3.ign.com/articles/115/1153279p1.html (as of March 2011), 2011.

[Jennings-Teats 10] Martin Jennings-Teats, Gillian Smith & Noah Wardrip-Fruin. *Polymorph: dynamic difficulty adjustment through level generation*. In Proceedings of the 2010 Workshop on Procedural Content Generation in Games, PCGames '10, pages 11:1–11:4, New York, NY, USA, 2010. ACM.

[Johnson 04] W. Lewis Johnson, Carole Beal, Anna Fowles-Winkler, Ursula Lauper, Stacy Marsella, Shrikanth S. Narayanan, Dimitra Papachristou & Hannes Vilhjalmsson. *Tactical language training system: An interim report*. In Proceedings of the Conference on Intelligent Tutoring Systems (ITS), pages 336–345, Berlin, Germany, June 2004.

[Kallmann 98] Marcelo Kallmann & Daniel Thalmann. *Modeling Objects for Interaction Tasks*. In Proceedings of the Eurographics Workshop on Animation and Simulation, pages 73–86, 1998.

[Kazmi 10] S. Kazmi & I.J. Palmer. *Action Recognition For Support Of Adaptive Gameplay: A case study of a first person shooter*. International Journal of Computer Games Technology, 2010.

[Kessing 12] Jassin Kessing, Tim Tutenel & Rafael Bidarra. *Designing semantic game worlds*. In PCG '12: Proceedings of the 2012 Workshop on Procedural Content Generation in Games, Raleigh, NC, USA, May 2012.

[Konami 07] Konami. *Pro Evolution Soccer 2008*, 2007.

[Lager 09] Craig Lager. *Adam Atomic on Canabalt*. Gaming Daily, September 2009.

[Lampton 05] Donald R. Lampton, James Bliss & Glenn A. Martin. *Performance Measurement and Training Feedback in a Military Collaborative Virtual Environment*. In First International Conference on Virtual Reality, July 2005.

[Lane 07] H. Chad Lane, Mark Core, David Gomboc, Arshish Karnavat & Milton Rosenberg. *Intelligent tutoring for interpersonal and intercultural skills*. In Proceedings of the Interservice/Industry Training, Simulation and Education Conference (I/ITSEC 2007), pages 1–11, 2007.

[Latoschik 05] Marc Erich Latoschik, Peter Biermann & Ipke Wachsmuth. *Knowledge in the Loop: Semantics Representation for Multimodal Simulative Environments*. In Proceedings of the 5th International Symposium on Smart Graphics 2005, pages 25–39, Munich, Germany, August 2005.

[Liapis 13] Antonios Liapis, Georgios N. Yannakakis & Julian Togelius. *Sentient Sketchbook: Computer-Aided Game Level Authoring*. In Proceedings of ACM Conference on Foundations of Digital Games, 2013.

[Lopes 10] Ricardo Lopes, Tim Tutenel, Ruben M. Smelik, Klaas Jan de Kraker & Rafael Bidarra. *A Constrained Growth Method for Procedural Floor Plan Generation*. In Proceedings of GAME-ON 2010, the 11^{th} International Conference on Intelligent Games and Simulation, Leicester, UK, November 2010. EUROSIS.

[Lopes 11a] Ricardo Lopes & Rafael Bidarra. *A Semantic Generation Framework for Enabling Adaptive Game Worlds*. In ACE '11: 8^{th} International Conference on Advances in Computer Entertainment Technology, New York, 2011. ACM.

[Lopes 11b] Ricardo Lopes & Rafael Bidarra. *Adaptivity challenges in games and simulations: a survey*. IEEE Transactions on Computational Intelligence and AI in Games, vol. 3, no. 2, pages 85 –99, june 2011.

[Lopes 12] Ricardo Lopes, Tim Tutenel & Rafael Bidarra. *Using gameplay semantics to procedurally generate player-matching game worlds*. In PCG '12: Proceedings of the 2012 Workshop on Procedural Content Generation in Games, Raleigh, North Carolina, USA, 2012. ACM.

[Lopes 13a] Ricardo Lopes, Ken Hilf, Luke Jayapalan & Rafael Bidarra. *Gameplay semantics for authoring adaptivity in mobile games*. In Poster Proceedings of the 8^{th} International Conference on the Foundations of Digital Games (FDG 2013), Chania, Crete, Greece, May 2013.

[Lopes 13b] Ricardo Lopes, Ken Hilf, Luke Jayapalan & Rafael Bidarra. *Mobile adaptive procedural content generation*. In Proceedings of the fourth workshop on Procedural Content Generation in Games (PCG 2013), Chania, Crete, Greece, May 2013.

[Magerko 06] B. Magerko, B. Stensrud & L. Holt. *Bringing the schoolhouse inside the box - A tool for engaging, individualized training*. In Proceedings of the 25th Army Science Conference. ASC, November 2006.

[Magerko 08a] Brian Magerko. *Adaptation in Digital Games*. IEEE Computer magazine, vol. 41, no. 6, pages 87–89, June 2008.

[Magerko 08b] Brian Magerko, Carrie Heeter, Joe Fitzgerald & Ben Medler. *Intelligent adaptation of digital game-based learning*. In Future Play '08: Proceedings of the 2008 Conference on Future Play, pages 200–203. ACM, 2008.

[Martin 10] Glenn A. Martin, Charles E. Hughes, Sae Schatz & Denise Nicholson. *The use of functional L-systems for scenario generation in serious games*. In PCGames '10: Proceedings of the 2010 Workshop on Procedural Content Generation in Games, pages 6:1–6:5. ACM, 2010.

[Maxis 08] Maxis. *Spore*, 2008.

[McCrae 09] James McCrae & Karan Singh. *Sketch-based Path Design*. In GI '09: Proceedings of Graphics Interface 2009, pages 95–102, Toronto, Ontario, Canada, 2009. Canadian Information Processing Society.

[Miller 95] George A. Miller. *WordNet: A Lexical Database for English*. Communications of the ACM, vol. 38, pages 39–41, 1995.

[Missura 09] Olana Missura & Thomas Gärtner. *Player Modeling for Intelligent Difficulty Adjustment*. In Proceedings of the ECML–09 Workshop From Local Patterns to Global Models (LeGo–09), Bled, Slovenia, 2009.

[Mott 06] Bradford W. Mott & James C. Lester. *U-director: a decision-theoretic narrative planning architecture for storytelling environments*. In Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems, pages 977–984, New York, NY, USA, 2006. ACM.

[Müller 06] Pascal Müller, Peter Wonka, Simon Haegler, Andreas Ulmer & Luc Van Gool. *Procedural Modeling of Buildings*. In SIGGRAPH '06: Proceedings of the 33^{rd} Annual Conference on Computer Graphics and Interactive Techniques, pages 614–623, Boston, MA, USA, July-August 2006. ACM.

[Newell 08] Gabe Newell. *Gabe Newell Writes for Edge*, 2008.

[Niehaus 09] James Niehaus & Mark O. Riedl. *Scenario Adaptation: An Approach to Customizing Computer-Based Training Games and Simulations*. In Proceedings of the AIED 2009 Workshop on Intelligent Educational Games, pages 89–98, 2009.

[Nielsen Company 09] Nielsen Company. *Insights on Casual Games*, September 2009.

[Nintendo EAD 08] Nintendo EAD. *Mario Kart Wii*, 2008.

[Nitsche 06] Michael Nitsche, Calvin Ashmore, Will Hankinson, Robert Fitzpatrick, John Kelly & Kurt Margenau. *Designing Procedural Game Spaces: A Case Study*. In FuturePlay 2006 Proceedings, October 2006.

[Olesen 08] Jacob Kaae Olesen, Georgios N. Yannakakis & John Hallam. *Real-time challenge balance in an RTS game using rtNEAT*. In IEEE Symposium On Computational Intelligence and Games, 2008 (CIG '08), pages 87–94, 2008.

[Pedersen 10] Chris Pedersen, Julian Togelius & Georgios N. Yannakakis. *Modeling Player Experience for Content Creation*. IEEE Transactions on Computational Intelligence and AI in Games, vol. 2, no. 1, pages 54–67, 2010.

[Peirce 08] N. Peirce, O. Conlan & V. Wade. *Adaptive Educational Games: Providing Non-invasive Personalised Learning Experiences*. In Digital Games and Intelligent Toys Based Education, 2008 Second IEEE International Conference on, pages 28–35, Nov. 2008.

[Peters 03] Christopher Peters, Simon Dobbyn, Brian MacNamee & Carol O'Sullivan. *Smart Objects for Attentive Agents*. In Proceedings of the International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG '03), pages 1–8, Plzen - Bory, Czech Republic, February 2003.

[Peytavie 09] Adrien Peytavie, Eric Galin, Jérome Grosjean & Stéphane Merillou. *Arches: a Framework for Modeling Complex Terrains*. In Computer Graphics Forum: Proceedings of Eurographics 2009, pages 457–467. Eurographics Association, 2009.

[Pita 07] James Pita, Brian Magerko & Scott Brodie. *True story: dynamically generated, contextually linked quests in persistent systems*. In Future Play '07: Proceedings of the 2007 conference on Future Play, pages 145–151, New York, NY, USA, 2007. ACM.

[Quantic Dream 10] Quantic Dream. *Heavy Rain*, 2010.

[Ramirez-Cano 10] Daniel Ramirez-Cano & Simon Colton. *Player classification using a meta-clustering approach*. In Proceedings of the 3rd Annual International Conference Computer Games, Multimedia & Allied Technology, pages 297–304, 2010.

[Rani 05] P. Rani, N. Sarkar & C. Liu. *Maintaining Optimal Challenge in Computer Games through Real-Time Physiological Feedback*. In 11th International Conference on Human Computer Interaction, pages 184–192. Lawrence Erlbaum Associates, Inc, 2005.

[Raybourn 07] Elaine M. Raybourn. *Applying simulation experience design methods to creating serious game-based adaptive training systems*. Interacting with Computers, vol. 19, no. 2, pages 206–214, 2007.

[Remedy Entertainment 01] Remedy Entertainment. *Max Payne*, 2001.

[Roberts 08] David L. Roberts & Charles L. Isbell. *A Survey and Qualitative Analysis of Recent Advances in Drama Management*. International Transactions on Systems Science and Applications, vol. 4, no. 2, pages 61–75, 2008.

[Rowe 09] J. Rowe, B. Mott, S. McQuiggan, J. Robison, S. Lee & J. Lester. *Crystal Island: A Narrative-Centered Learning Environment for Eighth Grade Microbiology*. In Proceedings of the AIED'09 Workshop on Intelligent Educational Games, pages 11–20, 2009.

[Rowe 10a] J. Rowe & J. Lester. *Modeling User Knowledge with Dynamic Bayesian Networks in Interactive Narrative Environments*. In Proceedings of the Sixth Annual AI and Interactive Digital Entertainment Conference, pages 57–62, 2010.

[Rowe 10b] Jonathan P. Rowe, Lucy R. Shores, Bradford W. Mott & James C. Lester. *Individual differences in gameplay and learning: a narrative-centered learning perspective*. In Proceedings of the Fifth International Conference on the Foundations of Digital Games, pages 171–178, New York, NY, USA, 2010. ACM.

[Shaker 10] N. Shaker, J. Togelius & G. N. Yannakakis. *Towards Automatic Personalized Content Generation for Platform Games*. In Proceedings of the Sixth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE), pages 63–68, October 2010.

[Shaker 12] Noor Shaker, Georgios N. Yannakakis, Julian Togelius, Miguel Nicolau & Michael O'Neill. *Evolving Personalized Content for Super Mario Bros Using Grammatical Evolution*. In Proceedings of Artificial Intelligence and Interactive Digital Entertainment Conference, 2012.

[Sharma 07] Manu Sharma, Santiago Ontanon, Christina Strong, Manish Mehta & Ashwin Ram. *Towards Player Preference Modeling for Drama Management in Interactive Stories*. In Proceedings of the Twentieth International FLAIRS Conference (FLAIRS07), pages 571–576, 2007.

[Smelik 11a] Ruben M. Smelik. *A Declarative Approach to Procedural Generation of Virtual Worlds*. PhD thesis, Delft University of Technology, November 2011.

[Smelik 11b] Ruben M. Smelik, Tim Tutenel, Klaas Jan de Kraker & Rafael Bidarra. *A Declarative Approach to Procedural Modeling of Virtual Worlds*. Computers & Graphics, vol. 35, no. 2, pages 352–363, April 2011.

[Smelik 14] Ruben M. Smelik, Tim Tutenel, Rafael Bidarra & Bedrich Benes. *A survey on procedural modeling for virtual worlds*. Computer Graphics Forum, 2014. doi: 10.1111/cgf.12276.

[Smith 09] Gillian Smith, Mike Treanor, Jim Whitehead & Michael Mateas. *Rhythm-based level generation for 2D platformers*. In Proceedings of the 4th International Conference on Foundations of Digital Games, FDG 2009, pages 175–182. ACM, April 2009.

[Smith 11] Adam M. Smith, Chris Lewis, Kenneth Hullett, Gillian Smith & Anne Sullivan. *An Inclusive View of Player Modeling*. In 6th International Conference on Foundations of Digital Games, Bordeaux, France, July 2011.

[Sorenson 10] Nathan Sorenson & Philippe Pasquier. *Towards a Generic Framework for Automated Video Game Level Creation*. In Applications of Evolutionary Computation, EvoApplicatons 2010, Proceedings, vol. 6024, pages 131–140. Springer, April 2010.

[Spronck 06] Pieter Spronck, Marc Ponsen, Ida Sprinkhuizen-Kuyper & Eric Postma. *Adaptive game AI with dynamic scripting*. Machine Learning, vol. 63, pages 217–248, June 2006.

[Spronck 10] Pieter Spronck & Freek den Teuling. *Player Modelling in Civilization IV*. In Proceedings of the Sixth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE), pages 180–185, 2010.

[Sullivan 10] Anne Sullivan, Michael Mateas & Noah Wardrip-Fruin. *Rules of engagement: moving beyond combat-based quests*. In Proceedings of the Intelligent Narrative Technologies III Workshop, pages 11:1–11:8, New York, NY, USA, 2010. ACM.

[Thue 07] David Thue, Vadim Bulitko, Marcia Spetch & Eric Wasylishen. *Interactive storytelling: A player modelling approach*. In Proceedings of the third Artificial Intelligence and Interactive Digital Entertainment conference (AIIDE07), pages 43–48, June 2007.

[Togelius 07] J. Togelius, R. De Nardi & S.M. Lucas. *Towards automatic personalised content creation for racing games*. In IEEE Symposium on Computational Intelligence and Games, 2007. CIG 2007, pages 252–259, April 2007.

[Togelius 11] Julian Togelius, Emil Kastbjerg, David Schedl & Georgios N. Yannakakis. *What is procedural content generation?: Mario on the borderline*. In Proceedings of the 2nd International Workshop on Procedural Content Generation in Games, PCGames '11, pages 3:1–3:6, New York, NY, USA, 2011. ACM.

[Tognetti 10] Simone Tognetti, Maurizio Garbarino, Andrea Bonarini & Matteo Matteucci. *Modeling enjoyment preference from physiological responses in a car racing game*. In IEEE Conference on Computational Intelligence and Games, 2010. CIG 2010., pages 321–328. IEEE, August 2010.

[Tutenel 08] Tim Tutenel, Rafael Bidarra, Ruben M. Smelik & Klaas Jan de Kraker. *The role of semantics in games and simulations*. ACM Computers in Entertainment, vol. 6, pages 1–35, 2008.

[Tutenel 09a] Tim Tutenel, Rafael Bidarra, Ruben M. Smelik & Klaas Jan de Kraker. *Rule-based Layout Solving and its Application to Procedural Interior Generation*. In 3AMIGAS: Proceedings of the CASA 2009 Workshop on 3D Advanced Media in Gaming and Simulation, pages 15–24, Amsterdam, The Netherlands, June 2009.

[Tutenel 09b] Tim Tutenel, Rafael Bidarra, Ruben M. Smelik & Klaas Jan de Kraker. *Using Semantics to Improve The Design of Game Worlds*. In Proceedings of the Fifth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE), pages 100–105, 2009.

[Tutenel 11a] Tim Tutenel, Ruben M. Smelik, Ricardo Lopes, Klaas Jan de Kraker & Rafael Bidarra. *Generating consistent buildings: a semantic approach for integrating procedural techniques*. IEEE Transactions on Computational Intelligence and AI in Games, vol. 3, no. 3, pages 274–288, 2011.

[Tutenel 11b] Tim Tutenel, Roland van der Linden, Marnix Kraus, Bart Bollen & Rafael Bidarra. *Procedural filters for customization of virtual worlds*. In PCG '11: Proceedings of the 2011 Workshop on Procedural Content Generation in Games, Bordeaux, France, June-July 2011. ACM.

[Tutenel 12] Tim Tutenel. *Semantic Game Worlds*. PhD thesis, Delft University of Technology, December 2012.

[Valve Corporation 08] Valve Corporation. *Left 4 Dead*, 2008.

[Valve Corporation 09] Valve Corporation. *Left 4 Dead 2*, 2009.

[van der Linden 13a] Roland van der Linden, Ricardo Lopes & Rafael Bidarra. *Designing procedurally generated levels*. In Proceedings of the the second workshop on Artificial Intelligence in the Game Design Process., 2013.

[van der Linden 13b] Roland van der Linden, Ricardo Lopes & Rafael Bidarra. *Procedural dungeon generation (accepted for publication)*. IEEE Transactions on Computational Intelligence and AI in Games, 2013.

[van Est 11] Casper van Est & Rafael Bidarra. *High-level scenario editing for simulation games*. In Proceedings of the 6nd International Conference on Computer Graphics Theory and Applications - GRAPP 2011, 2011.

[Walker, J. 09] Walker, J. *Rock, Paper, Shotgun webzine: Left 4 Dead 2: Exclusive RPS Hands-On Preview*, 2009.

[Westra 09] Joost Westra, Hado Hasselt, Frank Dignum & Virginia Dignum. *Adaptive Serious Games Using Agent Organizations*. Agents for Games and Simulations: Trends in Techniques, Concepts and Design, vol. LNCS 5920/2009, pages 206–220, 2009.

[Westra 10] Joost Westra, Frank Dignum & Virginia Dignum. *Keeping the Trainee on Track*. In IEEE Conference on Computational Intelligence and Games, 2010. CIG 2010., pages 450–457. IEEE, August 2010.

[Yannakakis 09] G.N. Yannakakis & J. Hallam. *Real-Time Game Adaptation for Optimizing Player Satisfaction*. IEEE Transactions on Computational Intelligence and AI in Games, vol. 1, no. 2, pages 121–133, June 2009.

[Yannakakis 11] Georgios N. Yannakakis & Julian Togelius. *Experience-Driven Procedural Content Generation*. IEEE Transactions on Affective Computing, vol. 99, pages 147–161, 2011.

List of Figures

List of Tables

Acknowledgements

Dear reader, here we are, at the acknowledgments section, the most read section of any PhD thesis. There is a reason for that. All the science, all the work we devote our lives to are not really that important unless we share it with other human beings. Even in the most individual tasks (like a PhD in computer science) we seek and treasure validation, debate, discussion, appraisal, intrigue, team work or just gossip near the coffee machine. So it is natural that the people who went through all these motions (including myself) want to gather all around and celebrate all the accomplishments they achieve together. This section is that celebration.

For these reasons, it was natural for me to choose to mainly acknowledge the people who had a direct effect in this book (and not all the people who lead me to this point in my life). The latter would need many pages, many stories and, to be honest, a much more important introspection. To all my old and new friends, ancient and recent colleagues (including names from below), we will have many opportunities to celebrate past and future landmarks or just silly occasions. You know who you are and you know who I am.

So, let's turn to the people who put the words in this book, beside me. I'd foremost like to thank Rafael Bidarra, my co-promotor and daily supervisor. The first thing we did was fight together to get our funding, with patience and inspiration. So thank you for "our" proposal, our work together before we could start working. As for the remaining years, thank you for your daily supervision. We are some sort of counterparts, where I am an assertive pessimist and you an enthusiastic optimist, where I am a *"this will never work"* to your *"this is totally going to work"*. It was a pleasure and a success to be complemented by you.

I'd also like to thank my promotor, Elmar Eisemann, for his efficient, quick and always sharp review of my thesis. I would also like to thank you for jumping into this already moving train and for always being understanding but demanding. Furthermore, it is inspiring to watch you reinvigorate our group with your relentless passion and intelligence. So, extra thanks for having accepted the challenge of leading us!

I am also grateful to my committee members, for having accepted being part of this day, for their sharp comments and for their upcoming questions. And an extra thanks for having worked on this during your summer. My special thanks go to Erik

Jansen for being my first promotor, and helping me find my way, early on. I would also like to single out prof. Joaquim Jorge for having been an integral part in this Delft adventure. Thank you not only for being in my committee, but mostly for having pointed me in this direction back in 2008. And, of course, thank you for putting in a good word for me to Rafael.

I would like to thank the Delft graphics crew, which have filled all these years with all the adventures I mentioned in the first paragraph. I was fortunate enough of having been a part of almost two different groups (and its transition). I would like to thank the ”old school crew” for having been my exciting start in this research world and in Delft, in particular Erik, Wim, Frits, Peter van Nieuwenhuizen, Charl, Gerwin, Ruud, Bart, Matthijs, Jassin, Fernando, Joel, Xin, Ruben, Tim (yeah, Ruben, Tim, you are old school, deal with it!). I think the transition between old school and new school is THE place to be, so thank you Rafa, Francois, Noeska, Thomas, Peter Kok, Christian. I am also grateful to the ”new school crew”, of course, for having the ability of immediately and smoothly materializing into an awesome group. So thank you, Changgong (see, you are first!), Elmar, Anna, Bert, Matthias, Jingtang, Timothy, Jean-Marc, Ben, Renata, Cees-Willem, Berend, Sergio and Pedro. I also thank my students, Antony, Shiyang, Roland and Marlon for helping me and inspiring me with your creativity. Uff, I look back into this list of names... good times! ☺

Furthermore, I would like to single out my paranymphs, Ruben and Tim, for being my partners in crime. Thank you for sitting beside me, both literally and metaphorically (ok, Tim, you still do that every day... just leave me alone, ok?☺). I hope that in many years, we can look into this period of time and laugh with old conference and office stories. I am greedy, so I also thank my third unofficial paranymph, Sergio. You already knew you were that before I asked you, right?

Of course, any rule is meant to be broken, so I extend my acknowledgments beyond. I thank my whole family, my wife, parents, brothers, grandparents, uncles and cousins, for having created this person who wrote this PhD thesis. As said before, there are many pages, many stories that transform this pride of mine into a pride of ours. Today, I remember the small things: thank you to my father for sharing a keyboard with a 10 year old hitting the *I,L,M* keys (invincibility cheat code) while he played *Wolfestein 3D*; thank you to my mother for never letting me quit anything, while I was growing up; thank you to Raul and Rodrigo for aspiring to beat me in most games and never doing so (who I am kidding, we know it is the other way around).

Finally, my biggest accomplishment between 2009 and 2014 was not this PhD. It was looking down the aisle while Luísa walked towards me. Everything I do only exists once I share with you.