Multi-dimensional Player Skill Progression Modeling for Procedural Content Generation

Francisco Fino Nina Pina Bicho

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisors: Prof. Carlos António Roque Martinho

Examination Committee
Chairperson: Prof. João António Madeiras Pereira
Supervisor: Prof. Carlos António Roque Martinho
Member of the Committee: Prof. Manuel Fernando Cabido Peres Lopes

Jun 2018
Acknowledgments

First and foremost, I want to thank my advisor, Professor Carlos Martinho for his insight, guidance, patience and support. I want also to thank my family, my parents and my sisters, for their immense support throughout the entire course. Finally, I want to thank Mariana Loureiro, for all her love and support and Diogo Garrett for his support and patience to review countless times everything I needed.
Abstract

Procedural Content Generation (PCG), i.e. how game content such as levels, items, obstacles and characters can be created algorithmically, is an increasingly important area and currently one of the most active topics within the games industry and game research. One of the crucial aspects of PCG is the capacity to maintain the player engaged and in flow. We explore how player skill progression could be used by PCG to create more appropriate challenges for each player and propose a model for content adaptation that takes this concept as its core feature.

Our approach introduces the player as an active element in the adaptation process and assumes both the player and the game should have an equal and active role in this process. Our adaptation explores how modeling the evolution of multiple dimensions of a same challenge while the game is played helps creating a better game experience for the player.

Our model was applied to a game entitled “Go, Go Hexahedron!”, developed to serve as a testbed game for our model. To evaluate our model, we present a validation process embedded in the game itself, with the purpose of providing a more direct and seamless way to analyze the players preference. The results of the evaluation of our approach in the context of an endless running side-scrolling platformer game revealed that players have consistent and specific preferences regarding how difficulty should evolve over the course of a game, which should be taken into account when designing an engaging game progression.

Keywords: procedural content generation; player adaptation; progression modeling; player modeling; player skill
Resumo

Conteúdo Gerado Procedimentalmente (PCG), isto é, a forma como o conteúdo de jogo como níveis, itens, obstáculos e personagens podem ser criados algorítmicamente, é uma área cada vez mais importante e atualmente um dos tópicos mais ativos na indústria e investigação de jogos. Um dos aspectos cruciais do PCG é a capacidade de manter o jogador envolvido e em flow. Neste trabalho, exploramos como a progressão da proficiência do jogador pode ser usada pelo PCG para criar desafios mais apropriados para cada jogador.

A nossa abordagem introduz o jogador como elemento ativo no processo de adaptação e assume que tanto o jogador como o jogo devem ter um papel igual e ativo neste processo. A nossa adaptação explora como a modelação da evolução de múltiplas dimensões do mesmo desafio enquanto um jogo é jogado ajuda a criar uma melhor experiência de jogo para o jogador.

O nosso modelo foi aplicado a um jogo chamado “Go, Go Hexahedron!”, desenvolvido para servir como jogo de teste para o nosso modelo. Para avaliar o nosso modelo, apresentamos um processo de validação embebido no próprio jogo, com o propósito de fornecer uma maneira mais direta de analisar as preferências dos jogadores. Os resultados da avaliação da nossa abordagem no contexto de um jogo de plataformas endless running side-scrolling revelou que os jogadores tem preferências específicas e consistentes em relação a como a dificuldade deve evoluir ao longo do jogo, que deve ser tomado em consideração na conceção de uma progressão de jogo envolvente.

Palavras-Chave: conteúdo gerado procedimentalmente; adaptação de jogador; modelação de progressão; modelação de jogador; proficiência de jogador
# Contents

## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>xi</td>
<td>xi</td>
</tr>
</tbody>
</table>

## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>xiii</td>
<td>xiii</td>
</tr>
</tbody>
</table>

## 1 Introduction

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 Motivation</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Problem</td>
<td>2</td>
</tr>
<tr>
<td>1.3 Hypothesis</td>
<td>3</td>
</tr>
<tr>
<td>1.4 Contributions</td>
<td>5</td>
</tr>
<tr>
<td>1.5 Outline</td>
<td>5</td>
</tr>
</tbody>
</table>

## 2 Related Work

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 Flow</td>
<td>7</td>
</tr>
<tr>
<td>2.2 Procedural Content Generation</td>
<td>8</td>
</tr>
<tr>
<td>2.3 Player modeling</td>
<td>10</td>
</tr>
<tr>
<td>2.4 Progression Modeling</td>
<td>13</td>
</tr>
<tr>
<td>2.5 Summary</td>
<td>18</td>
</tr>
</tbody>
</table>

## 3 Case Study

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1 Game Overview</td>
<td>21</td>
</tr>
<tr>
<td>3.1.1 Interface</td>
<td>22</td>
</tr>
<tr>
<td>3.1.2 Mechanics</td>
<td>23</td>
</tr>
<tr>
<td>3.1.3 Challenges</td>
<td>24</td>
</tr>
<tr>
<td>3.1.4 Rewards</td>
<td>31</td>
</tr>
<tr>
<td>3.1.5 Tutorial</td>
<td>31</td>
</tr>
<tr>
<td>3.2 Debug view</td>
<td>32</td>
</tr>
<tr>
<td>3.3 Summary</td>
<td>33</td>
</tr>
</tbody>
</table>

## 4 Progression Model

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1 Overview</td>
<td>35</td>
</tr>
<tr>
<td>4.2 Challenges Representation</td>
<td>36</td>
</tr>
<tr>
<td>4.3 Combos</td>
<td>37</td>
</tr>
<tr>
<td>4.4 Player skill</td>
<td>38</td>
</tr>
<tr>
<td>4.5 Skill-based Evolution</td>
<td>38</td>
</tr>
<tr>
<td>4.6 Reward System</td>
<td>39</td>
</tr>
<tr>
<td>4.7 Summary</td>
<td>40</td>
</tr>
</tbody>
</table>
List of Tables

6.1 Shapiro-Wilk Normality Test .................................................. 46
6.2 Wilcoxon Signed-Ranks Tests of the three segments (Plus and Minus Ratios) .... 49
6.3 Wilcoxon Signed-Ranks Tests of the three segments (Switch2plus and Switch2minus Ratios) .................................................. 49
# List of Figures

1.1 Screenshots of Pereira Thesis’s testbed game. The players play as a white cube that must overcome all the challenges thrown at them with the objective of surviving for the longest period of time possible. ................................................................................. 4

2.1 The flow channel - a narrow corridor between boredom and anxiety (Figure adapted from “Flow: The Psychology of Optimal Experience” [1]) .................................................................................................................... 8

2.2 Player Modeling: the core components (Figure adapted from [2]) .......................................................................................................................... 11

2.3 Screenshot of the Node Editor in action ............................................................................................................................. 15

2.4 Hierarchy of the node selection menu .................................................................................................................................... 16

2.5 Screenshot of a working progression graph ...................................................................................................................... 17

3.1 A screenshot of the game (first phase of the tutorial) ................................................................................................................. 21

3.2 Testbed game components enumerated (white lines and text) ......................................................................................... 22

3.3 The average and standard deviation of critical parameters ........................................................................................................ 23

3.4 Switch usage example on the dash-box challenge .............................................................................................................. 24

3.5 Challenges Composition Screen .................................................................................................................................. 24

3.6 Caption for LOF .................................................................................................................................................. 25

3.7 Skew Lines Challenge maximum growth dimension ........................................................................................................ 26

3.8 Cross Challenge. The player may use the double jump or slide mechanic in order to overcome it. .................................................................................. 26

3.9 Cross Challenge maximum growth dimensions .............................................................................................................. 27

3.10 Dash-Box Challenge. The player may use the double jump or dash mechanic in order to overcome it. .................................................................................. 27

3.11 Dash-Box Challenge maximum growth dimensions ........................................................................................................ 28

3.12 Jail Break Challenge. The player may use the slide or jump mechanic in order to overcome it. .................................................................................. 28

3.13 Jail Break Challenge maximum growth dimensions ........................................................................................................ 29

3.14 L challenge. The player may use the dash or slide mechanic in order to overcome it. .................................................................................. 29

3.15 L Challenge maximum growth dimensions .............................................................................................................. 30

3.16 Tobu challenge. The player may use the double jump or jump mechanic in order to overcome it. .................................................................................. 30

3.17 Tobu Challenge maximum growth dimensions ........................................................................................................... 31

3.18 Reward types: Green reward is 5 points, blue reward is 10 points, yellow reward is 15 points. .............................................................................................................. 31

3.19 Tutorial tooltips (keyboard). The game is slowed down in this phase, so the player can choose whenever to skip and maintain the bottom lane challenge, or switch and swap the bottom lane challenge with the top lane challenge and therefore choosing to face the top lane challenge. .............................................................................................................. 32

3.20 The debug view in the testbed game. .................................................................................................................................. 33
Chapter 1

Introduction

1.1 Motivation

Procedural Content Generation (PCG) [3, 4] is a growing trend in the software games industry and games research. Procedural content generation (PCG) is the programmatic generation of game content using a random or pseudo-random process that results in an unpredictable range of possible game play spaces [1]. PCG can be used to generate engaging and refreshing game worlds providing less predictable gameplay, as well as ensuring more replayability. PCG has been in games since the late 70’s with Beneath Apple Manor [2] preceding Rogue [3] released in 1980 and not only it is still relevant nowadays, it is actually growing in importance.

More games rely on PCG’s performance to provide a better player experience and a consistent replayability, that is, the ability of a game to present distinct content every time the game is played. Replayability plays a big role concerning the astonishing interest of PCG. Many of the most successful video games such as Minecraft [4] or Diablo III [5] use PCG in various ways (e.g. maps, characters, textures, inventory items) to ensure an appealing replayability.

Whereas the main purpose of PCG is to ensure replayability, the main aspect of level design is to provide the best player experience unwittingly disregarding replayability. In other words, as R. Kremer [6] describes, the basic purpose of level design is to interpret the game rules, and to translate them into a construct (a level) that best facilitates play. Level design is a core element in video game development and is necessary for two primary purposes: providing players with a goal [7] and providing players with enjoyable play experience.

In early days of video games, a single programmer would have to handle not only with the creation of the software itself but also with game design aspects such as game mechanics and as well as creating maps and layouts for the game. A time where a profession dedicated solely to level design did not exist. One of the first signs on video game history of mindful level design is from MUDs. An example of a well known MUD game is, for instance, MUD1 [6]. A MUD (Multi-User Dungeon) is a multiplayer real-time virtual world game, usually text-based. This was one of the first genres that required significant amounts of time to design engaging areas [7] in order to create a gamespace for players to dwell in.

The limitations of the standard level design, essentially, come from the lack of adaptation of the game

\[1\] http://pcg.wikidot.com/ (online as of 11/Mar/2018)
\[2\] D. Worth, 1978
\[3\] M. Toy and G. Wichman, 1980
\[4\] Mojang, 2000
\[5\] Blizzard Entertainment, 2012
\[6\] R. Bartle and R. Trubshaw, 1978
\[7\] https://en.wikipedia.org/wiki/Level_design (online as of 11/Mar/2018)
to the player. The inability to detect and correspondingly adapt may lead the player to become frustrated, or even bored and consequently disengaging the player. Every player has its own learning curve and because of this, the moment to display certain challenges might not be the best time for all players. A player becoming stuck for too long in a specific part is a simple example where it may lead the player to a frustrating experience. An illustration of a possible situation that potentially might lead to frustration is, for instance, a player getting stuck in a certain point of the game for the sole aspect of the game where he/she experiences difficulties. This approach usually tend to frustrate players needlessly.

Level design can be described as a layout of player progression i.e. how the player should move through the game and visualize progression. The majority of players may be familiar with the concept of player progression, even if only, at a subconscious level, that games should get harder over time. Most designers build increasing difficulty into successive levels, missions or worlds. But difficulty is only one aspect of the overall game experience. Player progression is the realized pattern of advance and the act of movement towards the ultimate goal (winning the game) that are crucial to an enjoyable experience for the player. Player Progression comes down to five key elements:

- Game Mechanics – the controls and interactions within the game, including, for instance, new weapons, abilities or powers.
- Experience Duration – the average time it takes to complete each stage, level, mission (including deaths if applicable) or course.
- Ancillary Rewards (visual, aural, decorative, etc.) – exciting environmental wonders, fancy visual effects and scripted events. An enjoyable game needs to have all the level, course or mission experiences built so that new visual rewards are empathized at a pace that keeps the user interested (in other words with an Environmental Progression in mind).
- Practical Rewards (gameplay relevant) – new game modes, upgrades and practical unlockable content are very useful enticing users to continue playing the game.
- Difficulty – not just how hard it is to pass obstacles and NPCs/bosses, but also how much risk is taken with respect to player injury/death, weapon depletion, or vehicle/equipment damage.

Games that do not structure the distribution of all these elements risk the danger of not keeping users engaged or overwhelming the player with too much.

### 1.2 Problem

While the current use of algorithms for automatically generating content has many advantages, the content itself is often random and not well suited to the player needs. In particular, it does not consider player skill progression, translating into either a boring gameplay, due to its simplicity, or a frustrating experience, due to an overwhelming difficulty (e.g. the difficulty spikes that break rogue-like experiences). The main issue with this approach is forcing the player to adapt entirely to the game, when it should be both the player and the game to adapt one to another.

The insufficient yet simple solution to this problem, is by offering configurable difficulty before starting the game. The first Call of Duty is an example of this approach. Nowadays, most games provide a slightly improved option which allow players to change the difficulty mid-game, for instance, The Elder Scrolls V: Skyrim. This option still limits the player to choose from a small discrete range of difficulty levels. Although the number of difficulties greatly varies from game to game, for instance, Diablo III is an extreme example containing 17 different difficulty levels, but in this case, it is an attempt to motivate the player to choose a higher difficulty level, rewarding the player with more gold, equipment and so on.

---

8Infinity Ward, 2003
9Bethesda Game Studios, 2011
Each person has a different gaming background, a different initial skill set as well as a different learning curve.

We argue that in order to improve player experience, difficulty should not be discretized into categories nor it should be used as a global setting affecting the whole game. Such an approach does not take into consideration that the player could be better at overcoming certain challenges with certain mechanics than others. The adaptation process should be continuously adjusted while the game is being played, providing challenges aligned with the evolution of the distinct skills of the player. It is important to note that players can take advantage of this, exploiting the game. Therefore, the game should not adapt blindly to the player, (like in The Elder Scrolls IV: Oblivion\textsuperscript{11}) but instead design mechanisms to encourage players to evolve.

Concretely our research question is: how should a game with a progressive increasing difficulty adapt itself in real time to the player to enhance the player experience, in particular adapting the difficulty increase based on the player skill while encouraging players to evolve?

1.3 Hypothesis

A good game is a game that provides the player a good experience \cite{8}. A good experience generates a sense of pleasure, satisfying the players need. Game design principles are used to guide game designers to increase the chance of successfully provide a good experience to the players. Two essential dimensions of play in game design are player skill and challenge. As described by Csikszentmihalyi \cite{1}, to maintain a person in flow, the play activity must reach a balance between the challenges provided by the game and the skill of the player. If the player’s perceived skill is much higher than the perceived difficulty of a challenge, the game becomes boring. On the other hand, if the player’s perceived skill is much lower than the perceived difficulty of the challenge, the activity turns into a frustrating experience, provoking anxiety. To our knowledge, most games do not usually consider skill in a flexible way in light of what is happening when the player is progressing through the game, i.e., there is no adaptation by the game regarding the skill of the player.

One example of video-game genres that we think there is room to improve, in terms of game adaptation considering skill are endless running games. Endless running games are platform games in which the player character is continuously moving forward through a usually procedurally generated, theoretically endless game world. Usually the game controls are limited to making the character jump, attack, or perform special actions. The objective of these games is to get as far as possible before the character dies\textsuperscript{12}. The way most endless running games raise difficulty over time, is either by introducing harder challenges or by increasing the challenges’ pace after a certain amount of time or traveled distance regardless of the player’s skill or progression. Video games like Subway Surfers\textsuperscript{13} and Temple Run\textsuperscript{14} are examples of this.

It is important to understand that a same “overall” skill could mean a different proficiency level at different tasks for different players, and that these different proficiencies will also evolve differently over the course of the game. Treating difficulty as one, overall, dimension can quickly frustrate players. A player could spend most of her time trying to overcome a specific challenge that requires a good grasp of specific mechanics, even though she is very proficient at all the remaining challenges and mechanics offered by the game. As such, we think that each challenge and the mechanics used to overcome it should have their own difficulty taken into account when thinking about game adaptation, rather than

\textsuperscript{11}Bethesda Game Studios, 2006
\textsuperscript{12}https://en.wikipedia.org/wiki/Platform_game#Endless_running_games (online as of 02/Mar/2018)
\textsuperscript{13}Kiloo and SYBO, 2012
\textsuperscript{14}Imangi Studios, 2011
having a difficulty that covers the whole set of challenges and all manners of overcoming said challenges.

Our approach will be to ensure that the adaptation process will treat difficulty relative to each challenge and the mechanics used to overcome it instead of using a global difficulty that affects almost every aspect of the game. For example, a player may be very proficient at overcoming hole challenges by jumping over them (demonstrated in figure 1.1a) even though he is not so good at sliding under the slide-wall challenges (as shown in figure 1.1b). Considering this situation, the next time the game adapts, it is not very reasonable to present both challenges at a higher and equal difficulty. Instead the difficulties should vary matching the player’s skill i.e. since the player was better jumping over the hole challenge, its difficulty should be higher than the sliding under the slide-wall where the player was significantly worse. This way the player is also encouraged, in a more controlled manner, to adapt and evolve in the challenge/mechanics he is not good at (in the example the sliding/slide-wall dimension). The player’s progress will be continuously monitored so performance prediction can inform the adaptation process.

In this work, we will model player progression in a game taking into consideration both dimensions, skill and challenge. We intend to keep the player in the flow zone where negative feelings (mostly anxiety and boredom), that produce “psychic entropy” [9] in the mind, would not occur. Our work will not adapt blindly to the player, or else the player would not feel challenged as the game would be too easy i.e. it would not be necessary for the player any type of attempt of adapting to the game. We propose that the answer to our problem may rely on a deliberately use of PCG and level design principles taking also into account the player skill.

Summarizing, our hypothesis is that difficulty cannot be viewed as a global setting affecting the whole game, difficulty should be divided into various dimensions since each player’s proficiency varies in each of these dimensions, adapting to each player differently; the player expresses preference for certain dimensions in detriment of others, considering the context of a game with a progressive increasing difficulty.

To illustrate our hypothesis, we will apply our model in a largely modified version of Pereira’s thesis testbed game [10], a side-scrolling endless-running platform game with various mechanics and challenges, displayed in figure 1.1.

![Figure 1.1: Screenshots of Pereira Thesis's testbed game. The players play as a white cube that must overcome all the challenges thrown at them with the objective of surviving for the longest period of time possible.](image)
1.4 Contributions

The contributions of this work start with the summary of the state of the art in research on the context of PCG, user modeling and progression modeling, where we analyze recent works and draw comparisons with our work. Additionally we detail the term flow which serves, in a way, as a foundation of our work. In this work, we explore how the dimensions of flow (i.e. player skill and challenge) could potentially guide PCG to present content more adapted to each player skill progression in order to provide an enjoyable and engaging experience to the players.

One of the most fundamental contributions is the development of a multi dimensional adaptation model in a context of a video game that procedurally generates content based on the player skill progression, where both the game and the player are active elements in the adaptation process. This model aims to contribute to the research area of progression modeling.

Another contribution is the implementation of a custom endless runner side-scrolling platform game titled “Go, Go Hexahedron!” where the progression model was applied serving as a case study where we can test the effectiveness of our model. Moreover, the implementation of a server to support automatic retrieval of telemetry of the game logs while players played the game.

Furthermore, to evaluate our approach, we present a novel validation process with the purpose of providing a more direct and seamless way to analyze player preference. Instead of interviewing players extensively about their experience, for instance, relative to two different versions of the game with different model settings, we embedded, in a complementary manner, the model validation process in the game itself. We think that by providing the player with the option to choose between multiple challenges, the one the player prefers to attempt is a much more automatic process to assess alternatives in a quantitative perspective.

Lastly, this work was submitted and posteriorly accepted, in the form of a full paper, to be featured in the International Conference on the Foundations of Digital Games (FDG). FDG is a major international event where the goal of the conference is the advancement of the study of digital games. The proceedings of the conference are archived in the ACM Digital Library.

1.5 Outline

The following chapters of this document are organized as following: chapter 2 begins by clarifying the term flow, since it will be a core concept used on our solution. We will then analyze relevant contemporary work concerning the scientific foundations of digital games, mostly emphasizing areas such as PCG, user modeling and progression modeling. In chapter 3, Case Study, we describe the game “Go, Go Hexahedron!” and its game elements such as mechanics, challenges, rules etc. Afterwards, in chapter 4, Progression Model section, we will detail the progression model, an overview of what it consists, and how it is connected to the game. In chapter 5, Evaluation, we explain how the tests were in fact done, i.e. the user tests procedure, as well as, how and what did we logged. Chapter 6, Results, presents the statistical analysis of our model regarding the case study. Lastly, chapter 7 concludes the document by summarizing it and identifying potential possible areas for future work.
Chapter 2

Related Work

Our work mainly relies on PCG and level design principles considering the player skill progression and a balanced player-game mutual adaptation. We will be analyzing recent works presenting an overall perspective regarding the research areas of procedural content generation, player modeling, as well as, progression modeling in video games that serve as base and inspiration in our work.

Thus, in order to simplify the overall organization, this section is divided into four categories: Flow (where we will explain in more detail the concept of flow as described by the psychologist M. Csikszentmihalyi [1]), Procedural Content Generation, Player modeling and Progression modeling.

2.1 Flow

The term flow, named by the psychologist M. Csikszentmihalyi [1], has been widely referenced across a variety of fields. In the video-game industry, more and more companies are taking the concept of flow into account when creating their games to improve player engagement.

In positive psychology, flow is the a mental state of heightened focus and immersion in activities such as art, play and work. A sense that one’s skills are adequate to cope with the challenges at hand. Concentration is so intense that there is no attention left to think about anything irrelevant, self-consciousness disappears momentarily and the sense of time becomes distorted.

Figure 2.1 illustrates better how to achieve flow. The two most important dimensions of the experience, challenges and skills, are represented on the two axes of the graph. A1, A2, A3 and A4 represents different states that one can possibly be.

Exemplifying this using video games, for instance A can be a person playing a platform game. The A1 state is when the player initiating and has practically no skills in the regarding game and the challenges presented are easy but appropriate to the person’s skill level, therefore the person is in the flow channel. The A2 state is when the player’s skill level increase but the challenges presented are still the same difficulty than before, the player leaves the flow channel and enters an undesired boredom state. On the other hand, if the challenges presented are too difficult for the player skill, the player becomes anxious, frustrated (A3). The A4 is achieved by either increasing the challenges’ difficulty to accommodate the player’s skill (A2 to A4) or the player improving in the game (e.g. mastering a certain game’s mechanic) allowing the player to still feel challenged with the current challenges and therefore reaching the flow channel again.

In our work, the concept of flow is specially important, because supports the idea of creating a adaptive game content considering the player skill which we will present. Our approach will connect both dimensions of flow, challenge and skill. By measuring the player skill we can present more appropriate
challenges, that more closely match the player skill, with the purpose of reaching and maintaining players in the flow channel.

2.2 Procedural Content Generation

Procedural content generation (PCG) refers to software that can create game content algorithmically as opposed to manually. PCG is widely used in games to essentially to ensure more replayability. In this chapter, we address an important taxonomy of PCG by Togelius et al.[11]: Procedural Content Generation in Games, A Textbook and an Overview of Current Research.

Togelius et al. [4, 11] present a detailed overview of recent research on procedural content generation regarding many different aspects such as dungeon generation [12], grammars and L-systems with applications to vegetation [13] and search-based approaches [14]. They provide a taxonomy that structures and highlights the differences and similarities between the approaches. The authors start by defining key concepts of this field followed by some examples of games, outlining desirable properties.

PCG, as the authors define, is the algorithmic creation of game content with limited or indirect user input [3]. PCG refers to computer software that can create game content on its own, or together with one or many human players or designers. It is safe to assume that implementations of PCG methods are solutions to content generation problems. A content generation problem may be to generate new grass with low level of detail within 50 milliseconds or even to generate a truly original idea for a game mechanic after days of computing time. The required properties of a solution are different for each application. The common desirable properties of PCG solutions are speed, reliability, controllability, diversity (to avoid the content looking like it is all minor variations on a tired theme) and believability.

A taxonomy of PCG

Considering the variety of content generation problems and methods that are available now, it is essential to have a structure to highlight the differences and similarities between approaches. This subsection is a version of the taxonomy of PCG originally presented by Togelius et al [4].
Online versus offline
PCG techniques can be used to generate content online [15, 16], as the player is playing the game. These techniques allow the generation of endless variations, making the game infinitely replayable and opening the possibility of generating player-adapted content, or offline during the development of the game or before the start of a game session. The usage of PCG for offline content generation is notably useful when generating complex content such as environments and maps. Left 4 Dead [4] is an example of the use of online content generation that provides a dynamic experience for each player by analyzing player behavior and altering the game state accordingly. One of the most successful video-game using offline content generation (in particular for generating the game’s world), is the well-known Minecraft.

Necessary versus optional
It is possible to use PCG to generate necessary game content that is required for the completion of a level, or it can be used to generate auxiliary content that can be discarded or exchanged for other content. The main difference between necessary and optional content is that necessary content should always be correct, while the optional content does not need to be always rigorously correct. Necessary content can be the main structure of levels in Super Mario Bros. [17], or the collection of required items to progress to the next level. Optional content may be the generation of different weapons in first-person shooter games, for example, Borderlands [2].

Degree and dimensions of control
The generated content by PCG can be controlled in distinct ways. A random seed is one way to gain control over the generation space; another way is to use a set of parameters that control the content generation along a number of dimensions. Minecraft uses random seeds which means the same world can be regenerated if the same seed is used.

Generic versus adaptive
Adaptive content generation refers to the paradigm of PCG where content is generated taking player behavior into account, as opposed to generic, which does not take into account player behavior. Adaptive content generation is a personalized or player-centered content generation where the player’s interaction is analyzed and content is created based on a player’s previous behavior. The vast majority of commercial games use PCG in a generic way, while adaptive PCG has been receiving increasing attention in academia recently. Left 4 Dead is an example of the use of adaptive PCG, where an algorithm is used to adjust the pacing of the game based on the player’s inferred emotional intensity.

Stochastic versus deterministic
Deterministic PCG allows the regeneration of the same content given the same starting point and method parameters as opposed to stochastic PCG where recreating the same content is usually not possible. A deterministic approach was used to regenerate galaxies in Elite [3].

Constructive versus generate-and-test
In constructive PCG, the content is generated in one pass, as commonly done in roguelike games. On the other hand, generate-and-test PCG alternate generating and testing in a loop, repeating until a satisfactory solution is generated. Yavalath [18] is a two-player board game generated completely by a computer program using the generate-and-test paradigm.

---

1 Valve Corporation, 2008
2 Gearbox Software, 2009
3 D. Braden and I. Bell, 1984
Automatic generation versus mixed authorship

The automatic generation is the content generated exclusively performed by a computer with limited input from game designers, who usually tweak the algorithm parameters to better control the generated content. However, a new paradigm emerged that focused on incorporating the designer and/or player input through the design process i.e. the player or a human designer cooperates with the algorithm to generate the desired content. Tanagra \cite{19} is an example of a system where the designer draws parts of a 2D level and a constraint satisfaction algorithm is used to generate the missing parts while retaining playability.

Discussion

The discussed work, Togelius et al.\cite{11} introduce the field of PCG and its important concepts providing a taxonomy of the different types of PCG. According to Togelius et al.\cite{11} taxonomy, our work can be categorized mainly as a content generator that is online (analyzes player’s behavior and alters the game state on the fly), necessary (the generated content needs to be correct or else the player can’t progress) and adaptive (the content is generated taking player behavior into account).

2.3 Player modeling

Yannakakis et al. \cite{2} provide a complete view of player modeling, a high level taxonomy and a discussion of the key components of a player’s model. The work focuses on a taxonomy of approaches for constructing a player model, the available types of data for the model’s input and a proposed classification of the model’s output.

The primordial goal of player modeling and player experience research is to understand how the interaction with a game is experienced by individual players. One of the main aims of the study of players in game is the understanding of players’ cognitive, affective and behavioral patterns. Player modeling, as defined by the authors, is the study of computational means for the modeling of player cognitive, behavioral, and affective states which are based on data (or theories) derived from the interaction of a human player with a game.

Computational model

Player modeling is, primarily, the study and use of artificial and computational intelligence techniques for the construction of computational models of player behavior, cognition and emotion. The authors argue that by clustering the available approaches for player modeling, we are faced with either model-based or model-free approaches as well as potential hybrids between them. The core components of a player model are illustrated in figure \ref{2}.

- **Model-based (top-down) approaches:** In a model-based or top-down approach the player model is built on a theoretical framework. Initially a model is hypothesized to explain phenomena, followed by an empirical phase in which they experimentally determine to what extent the hypothesized models fit observations. Top-down approaches to player modeling might refer to emotional models derived from emotion theories. As an example, top-down approaches to player modeling might refer to emotional models derived from emotional theories such as cognitive appraisal theory \cite{20} and game specific interpretations of Csikszentmihalyi’s concept of Flow \cite{1}.

- **Model-free (bottom-up) approaches:** Model-free approaches refer to the construction of an unknown mapping (model) between (player) input and a player state representation. In these ap-
proaches, observations (e.g. player data) are collected and analyzed to generate models without a strong initial assumption on what the model looks like. Classification, regression and preference learning techniques adopted from machine learning or statistical approaches are commonly used for this type of model. In model-free approaches there are attempts to model and predict players actions and intentions [21], as well as, data mining efforts to identify different behavioral playing patterns within a game [22].

- **Hybrid approaches**: The hybrid approaches places somewhere between model-based and model free approaches. The vast majority of the existing works on player modeling may be viewed as combination of model-based and model-free approaches, containing elements of both approaches.

The model's input can be of three main types: (1) data gathered from player gameplay in a game environment; (2) data collected as bodily responses to game stimuli such as physiology and body movements; and (3) the game context which comprises of any player-agent interactions but also any type of game content viewed, played, and/or created. The player profile, which is static information of the player (e.g. personality and gender), can also be used to enhance a player model.

The model's output is usually a set of particular player states. These states can be represented as a class, a scalar (or a vector of numbers) that maps to a player state, such as the emotional dimensions of arousal and valence or a behavioral pattern or even a relative strength (preference).

**Active Learning for Player Modeling**

Shaker et al. [23] propose the use of active learning for player experience modeling. A dataset from hundreds of players playing Infinite Mario Bros. [24] is used as a case study to learn models of player experience through the active learning approach. Active Learning (AL) refers to a set of methods where the learning algorithm is allowed to select the data to learn from. The selection is based on the learning gain and therefore the main advantage of these methods is that they significantly reduce the amount of data needed for training. The results obtained suggest that only part of the dataset is required for the construction of accurate models, this indicates the potential of the method and its benefits in cases where obtaining the data is expensive or time, storage or effort consuming. In this paper, the authors focused
on modeling players’ reported experience of frustration and challenge as they are very important factors in game design. Both our work and Shaker et al. [23] work address the issue of online content generation where the data is collected as needed, with the goal of providing a better gameplay experience to the player.

Building a model that learns from a given set of labeled data is what traditional supervised machine learning algorithms usually do. On the contrary, an active learner starts with a small training set of labeled instances \( L \), and is given a degree of control that allows it to carefully choose instances from a large unlabeled pool \( U \) and ask an oracle \( Q \) (in this case the player) for labeling of those. The instances with highest uncertainty, \( U_m \), are identified through calculating the probabilities of the membership of each unlabeled sample to each class (a class in this case is one of the possible values of the feedback provided by the user). Algorithm 1 describes in more detail the active learning process used.

Predicting player preference can be considered to be one of the main tasks of AI in games. Building such models however requires collecting representative and meaningful player information. Most of the approaches used for this task use machine learning techniques that learn from player behavior datasets annotated with player experience tags. While accurate models can be constructed using such methods, the process of designing the data collection surveys and gathering the needed information is time and effort consuming. The use of active learning facilitates the construction of accurate models while reducing the number of play testers required for data collection.

This work’s results suggest that models of high accuracy can be built using smaller portions of the original dataset. This algorithm benefits in cases where obtaining the data (labeled) is expensive or time consuming. The authors built models for predicting reported frustration and challenge.

The authors argue that, this work is the first step in building a full framework for content personalization. The models constructed can be used online during the content generation process. The player’s behavior is then captured and their feedback is collected as needed. As the process continues, personalized models are constructed that effectively explore the player as he explores the game ultimately leading to a better gameplay experience.

### Algorithm 1: Active learning framework

\[
C = \text{build a classification model from the labeled data } L; \\
e = \text{classification error on the testing set}; \\
\text{while } U \text{ is not empty and } e > \text{threshold do} \\
\quad U_m = \text{get most uncertain instance from the unlabeled data } U \text{ using the classifier } C; \\
\quad L_n = \text{label } U_m \text{ using an oracle } Q; \\
\quad L = L \cup L_n; \\
\quad C = \text{build a classification model from the labeled data } L; \\
\quad e = \text{classification error on the testing set}; \\
\text{end}
\]

### Discussion

In the first presented work, Yannakakis et al. [2] propose an overview of the current state of player modeling with a high level taxonomy. The authors established a computation player model and described its core components.

With this work, we can conclude that our approach is classified as a hybrid approach since our solution is not entirely based on an emotional theory although we make use of the term flow defined by Csikszentmihalyi [1]. We also inferred, in terms of input, that our work will be using, essentially, the gameplay data to make the game adapt better to the player.
Shaker et al. [23] propose the use of active learning for player experience modeling. It is important to reinforce that, even though user modeling is a very extensive area with many different approaches, most of the time, the goal is the same: to provide a better player experience through analyzing how the interaction with a game is experienced by individual players. Although our work focus is to model player’s skill progression and Shaker et al. [23] work is to create models of the player’s experience (challenge, frustration), both models are intended to be used on an online content generation scheme, where the data is collected as needed, with the goal of providing a better gameplay experience to the player.

2.4 Progression Modeling

The Chemistry of Game Design by Daniel Cook [25] compares the actual state of Game Design to alchemy and the urge to begin establishing a more systematic approach analogous to what transformed alchemy into modern chemistry. It is argued that nowadays games are built through habit, guesswork and devotion to pre-existing form. Building a testable model of game mechanics creates new possibilities for game balancing, original game design and the broader application of game design to other fields.

The main focus of the many attempts to define games has been the mechanic elements of the game such as actions that the system allows the player to perform. This approach treats games as self-contained logical systems. The author emphasizes the need of a working psychological model of the player.

Player Model

A player model is described as the simple idea that the player is the entity that is driven, consciously or subconsciously, to learn new skills high in perceived value. The pleasure is consequence of successfully acquiring skills. The three main concepts are Skill, Drive to learn and Perceived value.

- **Skill**: A skill is a behavior that the player uses to manipulate the world. Skills are either conceptual, such as navigating a map, or physical like jumping or pounding in a nail with a hammer.

- **Drive to Learn**: The drive to learn refers to the human psychological aspects of playing as an innate to all human beings, it is instinctual. We start from playing with blocks or dolls as children to more intricate hobbies as adults. We experience joy through being rewarded for learning. The feeling that gamers define fun is derived from the act of mastering knowledge, skills and tools [26].

- **Perceived value**: It is the reason players pursue skills with high perceived value over skills with low perceived value. As humans, our impulses to engage into play are instinctual, because it provides us a safe opportunity to learn behaviors that may improve our life. The perception of value is more important than an objective measurement value.

Skill Atoms

The author describes skill atom as a self contained atomic feedback loop. Skill atoms are defined as mix of the basic ingredients of a game such as tokens, verbs, rules, aesthetics, etc. Each atom is composed of four main elements:

- **Action**: The player performs an action.

- **Simulation**: Based off the action, an ongoing simulation is updated. For instance, a door might open.
• **Feedback:** The game provides feedback to let the player know how the simulation has changed state.

• **Modeling:** The player absorbs the feedback and updates their mental models on the success of their action. If they feel that they have made progress, they feel pleasure. If they master a new skill or other tool, they experience an even greater burst of joy. If they feel that their action has been in vain, they will feel bored or frustrated.

While a game is played, each atom is often looped multiple times before the user understands what it teaches. Skill chains are when several skill atoms are linked together to form a directed graph. This skill chain can visually represent how the players learn the utility of each skill in the chain. Constantly linking more chains will result in a network that describes the entire game. A skill chain is a general notation that can be used to model pretty much any game. It has the ability to better describe the player experience instead of the mere mechanics of the game providing a richer description of the important moments that occur during gameplay. A skill chain provides useful information about the state of the player, at any point in time it is possible to retrieve information such as:

• **Mastered skills:** Skills that have been recently mastered;

• **Partially mastered skills:** Skills that the player is toying with, but has not yet mastered;

• **Unexercised skills:** Skills the player has yet to attempt.

• **Active skills:** Skills that the player is actively using.

• **Burned out skills:** Skill atoms that the players have lost interest in exercising. When a burnout happens early on the skill chain, large portions of the player’s potential experience might become inaccessible.

PCG has been widely used in video games mainly to ensure replayability, usually game developers tend to subconsciously remove the player’s skill out of the game generation parameters which causes games to hardly adapt to the player. Pedro Pereira [10] proposes a progression model for specifying the game’s progression based on the player’s mastery of different dimensions of the experience – mechanics, challenges and pacing – and guide the PCG of content. This model was implemented for an endless-running platform game with a configurable visual tool to allow level designers to specify their own game logic integrating the progression model.

The author details how game progression can be expressed in terms of the evolution of player mastery over time through a progression map connecting the different elements of play using boolean logic and mastery gates. The visual game design support tool provided an intuitive interface to create such a progression graph.

**Progression Model**

The model aims to evaluate the player’s skill level with the mechanics, challenges and paces of a game in order to provide mechanics, challenges and pace adequate to the player’s skill.

The author define mastery as a measure of the proficiency the player gained over the different dimensions of play. The model considers three dimensions of play: mechanics (the actions the player has at his or her disposal to overcome the game challenges); challenges and; pacing (the rate at which challenges are offered to the player). Each element from these dimensions is rated according to the following mastery levels, expanded from the original list by Cook [25]:
• **Uninitiated** Elements that were never shown to the player.

• **Initiated** Elements previously presented to the player that the player is still learning or struggling to overcome elements that were never shown to the player.

• **Partially mastered** Elements that the player is currently exploring but have not yet mastered.

• **Mastered** Elements that appeared several times and were successful overcome.

• **Burned out** Elements that the player has lost interest in exercising, because they appear frequently and the player easily overcomes them

• **Frustrated** Elements that the player has lost interest in exercising, because they appear frequently and the player rarely is able to overcome them.

The model consists of different types of atoms. Each atom has a different usage and a particular role. The atoms are the following: Mechanic atoms, which represent a specific mechanic in-game; Challenge atoms, to be able to represent the obstacles; Pace atoms which is the rate that the game will provide the challenges to the player; Mastery atoms that represent a condition considering the level of mastery of a player (e.g. if the player skill is above or equal to mastered).

**Node Editor**

The custom graph editor was developed in Unity to facilitate the construction of different graphs. The graph editor tool is connected to the custom testbed unity game while providing a real time visualization of the current active atoms. The tool allows a game designer to visually specify how the player skill will influence the progression of the different dimensions of play and guide the PCG process.

![Figure 2.3: Screenshot of the Node Editor in action](image)

The editor supports 10 different types of game nodes (or atoms) and 5 auxiliary nodes to implement more complicated logic for the transitions. Each one of these nodes have an internal Boolean value which is sent to the nodes connected to their output. Most nodes will directly output the value of its input (and internal value), while the remaining nodes have a more complicated internal logic and conditions that need to be resolved to output a Boolean value of True. The active nodes have a green outline to easily differentiate from the disabled nodes.

The nodes, represented in figure [2.4] are separated in two categories: Boolean Nodes and Game Nodes.
Boolean nodes are meant to allow the user to use Boolean operators like AND, OR and NOT. The power node always outputs true. This node is usually used to define which game nodes will be active at the start of the game. Finally the Memory node is a representation of the Boolean logic D Flip-Flop. Once its input is True it will always output True thereafter, even if its input changes to False.

Game nodes represent the atoms referred in the Progression model. The games nodes are Mastery node, Timer node, Mechanic node, Challenge node, pace node and pointer node. The timer and pointer nodes are essentially auxiliary nodes to provide more flexibility to the user when using this tool. The mastery node function is to keep track of the performance of the player in a certain dimension of the game and output an active signal when the performance is at, above or below a certain threshold, that is relevant for the game designer. The mechanics nodes define what actions and other mechanics are available in the game at a certain point for the player to use and keep track of the associated mastery. For each mechanic node there is a mechanic associated. The challenge nodes represent an atomic challenge used in the game and has only element associated: a challenge. When the node is active in the graph, the challenge is available to be used in the procedurally generation of content and the skill of the player in overcoming this challenge is tracked by the system. The pace node represents a pacing that characterizes the flow of content generated by the game for the player. The pace node is mainly identified by a label, and characterized by the frequency of content generation and the amount of content generated each time the content is generated by the game. Pace node tracks player mastery in the context of the pace.

Each game node have two inputs, a standard input and a blocker input, and one output. The standard input is identifiable by its green color in the editor. A typical node is active if its standard input is active. If active, the internal and specific logic of the node will define whether the node will output an active signal to all nodes connected to the output. The blocker input is identifiable by its red color in the editor. The blocker input, when active, disables the node internal logic, but allows its internal value to propagate to the connected nodes.

A simple example of a working progression graph made with this tool is illustrated in figure 2.5. In this example when the game starts, the nodes directly connected to the Power node will be active. The nodes with a green outline are active, in this case these nodes are: Jump Mechanic, Pace Spawner,
Figure 2.5: Screenshot of a working progression graph

Pace, Challenge Spawner and Challenge 1x1. When the mastery of the player increases to Partially Mastered in the Jump mechanic then the game will enable the Double Jump mechanic. Similar to this, when the mastery of the player increases to Partially Mastered in the 1x1 challenge then the game will enable the new 2x2 challenge and at the same time, through a pointer node, disables the 1x1 challenge i.e. from that moment on, the game will only present 2x2 challenges.

Discussion

Daniel Cook [25] describes a way of creating a testable model of video-game mechanics using the game actions available to the player as entities that can be grouped into skill chains. These skill chains can visually represent how the players learn the utility of each skill in the chain. The author claims that it is possible to model most games using only skill chains.

We think it is also important to note that, from our search on this particular topic, player progression, Cook’s work is one of the very few works that had attempted to model the player’s skill and progression. In our work we will try to adapt the game to the player considering the player’s skill progression. Therefore, it is essential to understand the different skills’ status (e.g. mastered, burnout, etc) a player can
have. Pedro Pereira [10] proposes a model for progression in video games, using some of the features like mechanics and challenges, alongside with the rate of success of a player in each component. This model increases the level of replayability in single-player platform games which consequently increases fun and engagement.

This work have a different importance since we used it partly as a base to our own. It served some support for similar features (e.g. the player's skill measuring algorithm), although it was largely modified to suit our needs.

The visual editor itself was not used, since our goal was not to create a visual configurable tool but instead to optimize the algorithms to the particular modified endless testbed game. Although many of the editor's internal logic could be used in a future work, for instance, the challenge rate, pace enabling/disabling mechanics.

2.5 Summary

This chapter covered recent works presenting an overall view regarding the research areas of procedural content generation, player modeling, as well as, progression modeling in video games. We start by describing flow, coined by M. Csikszentmihalyi [1], which refers to the mental state of heightened focus and immersion in activities such as art, play and work, a sense that one's skills are adequate to cope with the challenges at hand. This work is fundamental base to understand how to approach a problem such as how to measure the player skill and what dimensions to consider. Our approach connects both dimensions of flow, challenge and skill.

We continue by detailing a PCG work by Togelius et al. [11] where the authors provide a taxonomy that structures and highlights the differences and similarities between the approaches. According to this taxonomy, our work can be categorized as a content generator that is online (analyses player's behavior and alters the game state on the fly), necessary (the generated content needs to be correct or else the player cannot progress) and adaptive (the content is generated taking player behavior into account).

Then, relative to Player modeling, i.e. defined as the study of computational means for the modeling of player cognitive, behavioral, and affective states which are based on data (or theories) derived from the interaction of a human player with a game, we presented two papers. The first work by Yannakakis et al. [2] provide a structured overview of player modeling and present a discussion of the key components of a player's model. They detail the taxonomy of approaches for constructing a player model, the available types of data for the model's inputs and propose a categorization of the model's outputs. According to this work, our approach could be classified as a hybrid approach in which we will essentially be using gameplay data to make the game adapt better to the player. The second work by Shaker et al. [23] propose the use of Active Learning for player experience modeling. Active learning refers to a set of methods where the learning algorithm is allowed to select the data to learn from. While the goal of this work is to create models of the player's experience (challenge, frustration), and our work focuses on modeling the player's skill progression, both address the issue of online content generation where the data is collected as needed, with the goal of providing a better gameplay experience to the player.

Finally, we detail two works in Progression Modeling. The first work by Cook [25] describes a way of creating a testable model of game mechanics using the game actions available to the player as entities that can be grouped into skill chains. These skill chains can visually represent how the players learn the utility of each skill in the chain. Cook's work was one of the first works to attempt modeling the player's skill and progression. In our work, we will adapt the content to the player based on his skill progression. The second work by Pedro Pereira [10] proposes a model for progression modeling in video games.
using features like mechanics and challenges, alongside with the rate of success of a player in each one of the components. This work have a particular importance since we used it partly as base to our own. Although we share similar features, e.g. the way player’s skill is measured, we largely modified the approach to allow for online capture of player preference elicitation.
Chapter 3

Case Study

In this chapter, we will describe the testbed game that we developed and used as our case study, “Go, Go Hexahedron!”, and its components: the rules, objectives, challenges, movement mechanics as well as the interface.

3.1 Game Overview

“Go, Go Hexahedron!” is an endless running side-scrolling platform game with a minimalist aspect featuring a white cube as the player controllable character.

The white cube is continuously “running” towards the right of the screen and its objective is to obtain the most points possible while avoiding every obstacle in the way. These points are obtained staying alive (the longer, the more points the player can get), and by catching the different reward spheres. The testbed game can either be played using a regular keyboard or a standard controller (Xbox One/360, PS4, etc).

Figure 3.1: A screenshot of the game (first phase of the tutorial)
The testbed game’s main design goals are:

– Challenging: The player not only has to use its capacity of using the correct mechanics as well to master their timing to be able to successfully surpass the challenges. The Procedural content generation of challenges makes it even harder over time.

– Accelerated Rhythm: The game provides three pacing phases where the spawning rate for each challenge is gradually increased. The decision making of each challenge requires speed and skill.

– Choosing your challenge: Each time the player has to face a challenge, the player is able to choose from two different challenges, the one that is more appealing by the player’s interests. Both challenges have the same base form but evolve in different ways which is directly related to the players’ actions. This one is specially important element to our evaluation which is further explained in the chapter 5, Evaluation.

### 3.1.1 Interface

The main screen of the game’s interface is quite simple and standard according to other video games of the same genre in the industry. The game and its interface components are visible in figure 3.2.

![Testbed game components enumerated (white lines and text)](image)

Figure 3.2: Testbed game components enumerated (white lines and text)

At the bottom left corner of the figure, we have our player controllable character, which is a white plain cube. The player has initially three lives, these lives are represented as white squares and can be found at the top left corner. The score is displayed at the top of the figure, the score can be increased by simply staying alive, or catching rewards. These rewards are displayed in the middle of the figure represented by the spheres with a plus sign. To stay alive the player must avoid the incoming challenges performing certain mechanics, these challenges come from the right to the left of the screen. There are always two challenges (excluding the initial part of the tutorial), the top lane challenge and the bottom lane challenge shown also in figure 3.2. More detailed pictures of the game interface can be found in appendix B.
3.1.2 Mechanics

In “Go, Go Hexahedron!”, mechanics are actions available to the player. These mechanics must be used in order to progress in the game, overcoming challenges. Different mechanics are used to overcome different challenges. These mechanics can be freely and fluidly chained together providing a more consistent and dynamic environment for the player, creating a better game experience. For instance, after jumping, it is possible to slide in mid-air to better avoid a challenge. In other words, it's possible to combine these mechanics to enable the player to surpass challenges easier. These mechanics are shown in figure 3.3. There are four different mechanics:

- The **single jump** mechanic: This mechanic is activated by pressing the up arrow key and allows the player to jump a low height but also a long distance, floating in the air for a couple of seconds. The figure 3.3a illustrates this mechanic.

- The **double jump** mechanic: This mechanic is activated by pressing the up arrow key after the single jump, i.e. pressing the up arrow key twice. The double jump ascends higher than the single jump although it falls much faster to the ground comparing to the single jump. The figure 3.3b illustrates the double jump mechanic.

- The **dash** mechanic: By pressing the right arrow key, the cube performs a dash oriented to the right i.e. a swift and fast move which can be used to surpass the gray challenges. It can also be useful to rapidly overcome challenges, under or over them, avoiding them faster. The figure 3.3c illustrates the dash mechanic.

- The **slide** mechanic: The down arrow key can be pressed to activate this mechanic. The sliding mechanic squishes the Player controllable character (the white cube) reducing its own height. This is particularly effective to pass under challenges, or even to adjust the hitbox in a middle of a jump in order to pass a challenge (to avoid touching the challenge). The figure 3.3d illustrates this mechanic.

![Figure 3.3: The average and standard deviation of critical parameters: Region R4](image-url)
In addition to the base mechanics, the player can also use the “switch” key/button. Challenges enter through the right side of the screen and are always presented to the player in pairs, one challenge on the top lane and another on the bottom lane. The player will always have to overcome the challenge on the bottom lane. While the challenges are still a certain distance away from the player, they can be switched using the switch key/button. This allows the player to choose which challenge to overcome. Every time the player presses the switch button the two challenges will swap places in a slow mode fashion where the rest of the game stops whilst this change is occurring. It’s also possible to revert the switch in case the player changes its mind but only while the switch animation is playing, after the animation stops, the swap is final. An example of a switch can be found in figure 3.4.

![Switch usage example on the dash-box challenge](image)

(b) The dash-box challenge after the switch

**3.1.3 Challenges**

Each challenge is composed by using one of at least three types of blocks (displayed in figure 3.5) where each color represents a type:

- **Regular Blocks:** Brown colored. To overcome regular blocks the player has to avoid a frontal hit even though it's possible to land on top of it without losing a life.
– Dasher Blocks: Grey colored. There are two ways to overcome dasher blocks: dodging it like a regular block (passing above or below it) or passing through it using the dash mechanic.

– Danger Blocks: Red colored. Similarly to regular blocks the only way to surpass danger blocks are either by passing above or below it. The main difference resides in the fact that danger blocks do not allow the player to land on top of it. Thus, if a player at any time, touches this block anywhere, loses a life.

The game offers six different challenges. Each challenge was designed so it could be overcome in different ways and using different mechanics. Each of these challenges were designed in a way that it is possible to surpass them using one of either two mechanical ways, i.e. for each one of the challenges, there’s always two paths one can take to overcome it. As the game becomes more difficult, each challenge will evolve in distinct dimensions, each evolution requiring more precision from the player’s execution. The evolution of each challenge is dependent of the player’ performance. The challenges the player must overcome in order to survive are constituted by 6 different types:

**Skew Lines Challenge**

The Skew Lines Challenge is composed by a danger block and three dasher blocks. The skew lines can be surpassed using a jump passing between the danger and dasher blocks or by dashing through the dasher blocks. Its initial form is represented in figure 3.6. The difficulty evolution will either: increase the height of the danger block or increase the width of the dash block. The maximum growth dimensions are displayed in figure 3.7.

![Figure 3.6: Skew Lines Challenge. The player may use the dash or jump mechanic in order to overcome it.](image)

\[1\] The screenshots shown were taken on the tutorial. This is important to note since the reason of having a single challenge (on the bottom lane) and the tooltips is because the screenshots were taken on the tutorial.
Cross Challenge

The Cross challenge is composed by six regular blocks (two of them are hidden in its initial form). The challenge can be surpassed using a double jump, passing over it, or by sliding under it. Its initial form is illustrated in figure 3.8. The difficulty evolution will either: increase the height of the two upper regular blocks, or increase the height of the lower regular block. The maximum growth dimensions are displayed in figure 3.9.

Figure 3.8: Cross Challenge. The player may use the double jump or slide mechanic in order to overcome it.
Dash-Box Challenge

The Dash-box Challenge is composed by a single dasher block. It is possible to overcome this challenge by either using a double jump, passing over it, or by dashing through it. Its initial form is illustrated in figure 3.10. The difficulty evolution will either: increase the height of the dasher block, or increase the width of the dasher block. The maximum growth dimensions are displayed in figure 3.11.

Figure 3.10: Dash-Box Challenge. The player may use the double jump or dash mechanic in order to overcome it.
Jail Break Challenge

The Jail Break Challenge is composed by six danger blocks and one regular block. It is possible to overcome this challenge by either jumping, passing in the middle of the challenge, or by sliding under it. Its initial form is illustrated in figure 3.12. The difficulty evolution will either: increase the height of the regular block (downwards), or increase the width of the middle danger block. The maximum growth dimensions are displayed in figure 3.13.

Figure 3.12: Jail Break Challenge. The player may use the slide or jump mechanic in order to overcome it.
L Challenge

The L Challenge is composed by two blocks, one dasher block and one danger block. It is possible to overcome this challenge by either dashing through the dasher block, passing in the middle of the challenge, or by sliding under it. Its initial form is illustrated in figure 3.14. The difficulty evolution will either: increase the height of the dasher block (downwards), or increase the width of the dasher block. The maximum growth dimensions are displayed in figure 3.15.
Tobu Challenge

Tobu Challenge is composed by two blocks, one regular block and one danger block. It is possible to overcome this challenge by either jumping passing in the middle of both blocks or, passing above both by double jumping. Its initial form is illustrated in figure 3.16. The difficulty evolution will either: increase the height of the regular block (upwards), or increase the height of the danger block. The maximum growth dimensions are displayed in figure 3.17.
3.1.4 Rewards

The rewards are visually represented as spheres with a plus sign. The player may catch a reward to increase its score. Rewards come in three different colors: green, blue and yellow. When the player catches a reward it is awarded with 5 (green reward), 10 (blue reward) or 15 (yellow reward) additional points. When a reward spawns, the game selects randomly between the three categories. These rewards are represented in figure 3.18.

Players increase their score by overcoming challenges and staying alive. Additionally, they can collect rewards to increase their score. Rewards were added, on the one hand, to add some diversity to the game but, more importantly, to motivate players to push themselves while the game progressively gets more difficult.

3.1.5 Tutorial

The tutorial was created to initiate players to the game environment, even though is not fundamentally different from the regular game, some things are changed to provide an easier and cleaner introduction to the game. One of the main differences is that every time the player is presented a challenge, the game changes its game speed to appear as if the game was in slow motion, whereby time appears to be slowed down. Thus, offering the player more time to analyze the presented challenges and how to react accordingly. This slow motion state is held until the player decides to, either switch the current challenge with the top lane challenge, or, keep the bottom lane challenge by pressing the confirmation button. This slow motion state is specially important to avoid time being a factor influencing players’ choices, i.e., time could lead players to choose the undesirable challenge (e.g. In practice, it is possible that a certain player may not be able to press the switch button in time). When in this state a tool tip is also displayed to show the player which key they may press to either switch challenges or to keep the
bottom lane challenge. This tool tip vary depending on which input device the player is using (controller or keyboard). This is illustrated in figure 3.19.

The tutorial was designed to introduce players to every challenge starting in their easiest form, but also to gather some initial player data regarding their skill in each challenge to initialize our progression model. Because of this, the game requires the player to play at least one time the tutorial. Since this phase the progression model is not altering the challenges difficulties, rewards do not appear as well. More information about the tutorial can be found in appendix D.

Figure 3.19: Tutorial tooltips (keyboard). The game is slowed down in this phase, so the player can choose whenever to skip and maintain the bottom lane challenge, or switch and swap the bottom lane challenge with the top lane challenge and therefore choosing to face the top lane challenge.

3.2 Debug view

The testbed game provides a debug view providing information concerning the model. The debug view was developed to provide a visual real-time validation of the model, as a way to verify its functionality and dynamic corresponded to what we expected. The debug view was not available to players during the tests. This view is illustrated in figure 3.20.

This view is intended to analyze, in real time, information about how the challenges are being spawned, i.e. information concerning the modes (plus and minus, detailed in the Evaluation chapter) and the “pairs” < challenge, mechanics > which contain the attempts and the calculated success rate (detailed in the chapter 4). In figure 3.20 we can observe three main sections:

– Debug Keys: The white text at the left shows the keys available, only in debug mode, that can do various things such as spawn random or specific challenges, deactivate collisions (which basically means infinite lives), stop the automatic challenge spawning, change the camera projection (orthogonal or perspective) and change the camera zoom. At the bottom left, in white text as well, it is possible to see the time passed.

– Challenges Information: Located beneath the challenges (in this case in the middle of the screen, the two green texts), displays useful information relative to current challenges. The information is divided in 8 sections separated by the “|” character, containing: The challenge name, the challenge growth mode (plus or minus, detailed in the next chapter, Evaluation), the challenge current difficulty in each dimension and the actual size in Unity meters (in figure there are two associated dimensions, which correspond to the width and height respectively), the dimension increased, the
overall challenge difficulty (mean between the previous difficulties), the number of spawns occurred of the challenge in the current mode and the total challenge spawns (the plus mode spawns and the minus mode spawns). There is also two green and red lines which represent the beginning and the ending of the challenges.

– Pairs \langle \text{challenge, mechanics} \rangle \text{: The pairs are represented as colored squares as shown in figure 3.21. The squares are located in the top left of the debug view. The associated colour represents the value of the skill measurement calculation relatively to a particular pair, where red symbolizes a value close to 0, yellow represents a value around 0.5 and green a value close or equal to 1. By selecting one of these squares it is possible to obtain more detailed information, (see figure 3.22) such as the mechanics used by the player, the name of the challenge, the more recent attempts at overcoming the challenges and their success, and the calculated skill.}

3.3 Summary

In this section, we described the implemented testbed game that we used as our case study. “Go, Go Hexahedron!” is an endless running side-scrolling platform game with a minimalist design, featuring a white cube as the player controllable character which is continuously “running” towards the right of the screen where the cube’s objective is to obtain the most points possible while avoiding every obstacle in the way.

We provided a concise overview of the game including its rules, objectives and explained the interface components, e.g. the tooltips, lives, score, etc. Then, we proceeded to enumerate the available mechanics and show how they work in the game. These mechanics include the single jump, double
Figure 3.22: Selecting one of the colored squares (i.e. a pair), more detailed information can be obtained about this pair. Showing, in particular, the mechanics used, the challenge and the respective attempts.

jump, dash, slide and switch. Concerning the challenges, we provided an illustration for each challenge available in the game and detailed how they evolve. We described the different rewards types and its respective scores. Finally, we displayed the debug view which provide vital information regarding the validation of the model.
Chapter 4

Progression Model

Level Design [27] teaches us how game levels should ensure that a player learns the required skills before making the player face greater challenges to allow for a smooth game progression. Our main goal with our progression model is to provide appropriate challenges accommodating the distinct skills of the player, enhancing the overall player experience. To ensure this, our model will be consistently updating itself, in real time, based on the player collected data.

The cornerstone of our work is the balance between the player and the game itself. Both entities should adapt to each other in an equal manner. The player and the game should continuously evolve in order to maintain the player in flow. Our approach consists in considering difficulty in terms of challenges dimensions of the game, instead of making difficulty influence the game as a whole. In other words, increasing the game difficulty does not correspond to increasing the difficulty in every aspect of the game. Each player can progress differently in each skill within the game and therefore evolve in a different manner. The ideal is to increase difficulty continuously in an adequate manner to maintain the player challenged. Although it is important to avoid making the player feel frustrated for a possible unfair difficulty increase, in a particular section of the game where one can have less skill comparing with others sections. Considering the player skill progression it is possible to increase difficulty in a more appropriate manner.

For the game to adapt to the player skill progression, our model needs to capture how well the player is doing. Thus, the model starts by measuring the player skill. Instead of measuring the overall player skill, we started by defining skill in terms of mechanics and associated challenge. Distinguishing skill in terms of the mechanics chosen to overcome a certain challenge allows our model to have finer information regarding skill. For every attempt, the model saves if the player succeed or failed at the challenge, by how much, and the mechanics used to overcome the challenge.

4.1 Overview

Before delving into the details of the model, we provide a general overview. Our model’s fundamental objective is to increase the difficulty where the player skill is best. Each time the game presents a new challenge to the player, our model evolves this challenge. In particular, it increases the difficulty in one of the dimensions of the challenge, the dimension the player is more skilled at overcoming. To inform the selection of the dimension to evolve, our model maintains the player skill value for all pairs (mechanics and corresponding challenge) used by the player during the game.

Each challenge can be overcome using certain combinations of mechanical actions (combos) and these combos are associated to the different dimensions of the challenge, e.g. for the dash-box chal-
lenge (see Figure 4.1), the ‘dash’ combo is associated with an increase in the length of the challenge, while the ‘double jump’ combo is associated with an increase in the height of the challenge. Since each time a player overcomes or fails a specific challenge the model register this information, it can now search the data using the challenge name and each one of the combos. For every match, the model calculates the skill associated with overcoming the challenge with this combo based on the most recent attempts, then compares these calculations to find the dimension with the best skill. After this step, the model knows exactly which pair has the best skill and also which dimension it is associated to. Therefore, the model is now able to increase the challenge’s difficulty in this particular dimension.

The next subsections details the representation of challenges, combos, pairs, how the player skill is calculated, and the reason for including the reward system.

4.2 Challenges Representation

The model was designed to work with challenges that could be surpassed through the use of different mechanical combos. The player has, at least, two different paths to overcome every challenge. As an example, consider the “dash-box” challenge displayed in Figure 4.1. In this case, the player can choose between ‘dashing’ through the gray matter or jumping over it using ‘double jump’.

Each time a challenge spawns, its difficulty will increase on one of the available dimensions. In “Go, Go Hexahedron!”, the increase in difficulty is represented by modifying the size of certain core parts of the challenges. For instance, in the “dash-box” case, the model will increase its difficulty in two dimensions that correspond to two possible ways of overcoming it: for the ‘double’ jump path, the difficulty increases by increasing the height of the challenge, for the ‘dashing’ path, by increasing the width of the challenge.

Although, in this case, difficulty increases by changing the size of the challenge, our model is not limited to size, i.e, the requirement is only to have a minimum and maximum value associated with the lower and higher difficulty, and a form to interpolate between the two. As such, the values could represent how fast a slicing blade is, the number of objects in a challenge, etc.

In our testbed game, there are 6 challenges with two dimensions for evolving difficulty. There are, however, many more multiple mechanical ways for the player to overcome each challenge.

(a) Player opting to dash and catch a reward overcoming the dash-box challenge.
(b) Player opting to double jump and also catching a reward overcoming the dash-box challenge.

Figure 4.1: The two possible paths one can take to overcome the Dash-box challenge.
In “Go, Go Hexahedron!”, the value used for interpolation between the two extremes of a certain dimension of a challenge is \( D \), calculated by the following equation:

\[
D = \log_5(spawn + 1)
\]

(4.1)

where \( spawn \) represents the number of times this particular challenge was spawned in the current game. As such, for the first spawn of a particular challenge, the difficulty for a certain increased dimension would be \( \log_5(1+1) \) resulting in a 43% difficulty. The second spawn would be \( \log_5(2+1) \) resulting a 68% difficulty and so on. After 4 spawns (considering both the challenge and the dimension chosen were the same for the 4 spawns), the dimension will be at its maximum difficulty (100%).

Although our initial approach was to increase the difficulty linearly, the feedback we received from the preliminary evaluations showed that players wanted a faster progression of the challenge difficulty. The original evolution of the challenges was too subtle to be understood by all players. To allow players to more clearly understand the changes taking place in the game, we opted to use a logarithmic curve to increase the challenge difficulty.

### 4.3 Combos

In “Go, Go Hexahedron!”, it is possible to combine multiple mechanics when overcoming any type of challenge. A combo is a combination of mechanical actions. Each mechanical action is represented by a constant state. The combo detection mechanism works as follows: every time a player uses a mechanical action, the associated state is inserted into the combo. Every combo always starts with the idle state. The idle state refers to ‘doing nothing’, i.e., the player did not execute any action. The model uses this state to identify when the player did not have enough time to react to a challenge. Thus, the smallest possible combo is made of 1 state, the idle state, while the largest combo is composed of 5 states (the idle state plus the 4 mechanical actions). Our current implementation does not take into account the order of the mechanical actions in the sequence or the number of times a specific action was used in a combo.

The duration in which the mechanical combo is kept in memory (tuned through playtesting) is approximately 0.6 seconds. In other words, if a player does not use any other mechanic during this time, the combo is considered over and resets to the idle state. After this time, any mechanical action is considered part of a new mechanical combo.

Every challenge is limited by two planes: the left plane (represented as a green line in the debug orthographic view), and the right plane (represented as a red line in the same view). The left plane represents the beginning of the challenge, while the right plane denotes its end. A challenge is considered to be in progress when the player’s cube is between the left plane and the right plane. Success or failure information on the use of the combo in the context of this specific challenge is saved when the player either (1) crosses the left plane first and then the right plane without ‘colliding’ with the challenge (success), or (2) ‘collided’ with the challenge after having crossed the left plane, meaning that it failed the challenge. Colliding will mean different thing based on the type of the challenge (e.g. you can traverse a grey obstacle with ‘dash’ but not a red one).

The mechanical combo is only saved in the “pairs” < challenge, mechanics > when the player attempts to surpass any challenge. This is done by either, checking if the player trespassed the left plane and then the right plane without colliding, or if the player has passed the left plane and collided with the challenge, meaning that it failed the challenge.
4.4 Player skill

To compute player skill, we recorded, for each attempt at a challenge, a success rating, measuring how well the player performed. Our conceptual definition of success rating is the following: 0.0 means the player totally failed the challenge; 1.0 means that the player totally succeeded at the challenge; values in between represent how easy or hard the challenge was to overcome (e.g. based on how close to the obstacle the player landed).

These success ratings, that represent how good (or bad) the player is at surpassing a certain challenge with a set of mechanics, are stored under a key \(<\text{challenge, mechanics}>\) in a hashtable. The multi-dimensional key is constituted by the array of mechanical actions used (combo) and the identifier of the challenge. The value associated with this key is the list of objects containing the success rating of the most recent attempts at overcoming the challenge using this set of mechanics. In other words, every time a player attempts to overcome a challenge, a new pair \(<\text{challenge, mechanics}>\) is created (or an existing one updated if the key already exists) and the associated list updated with the most recent success rating.

The list associated with each pair \(<\text{challenge, mechanics}>\) represents a sliding “window” recording the success ratings of the most recent attempts. Every time the player attempts to overcome a challenge using a specific combination of mechanics the new success rating is inserted in the respective sliding “window”. In our case study, the sliding “windows” registered the 10 more recent attempts using a FIFO (First-In, First-Out) policy, i.e. in the 11th attempt, the oldest attempt is deleted to insert the new one.

4.5 Skill-based Evolution

Each challenge is defined by the number of dimensions (in our case study, the challenges had 2 dimensions for evolution), the minimum and maximum values of each dimension, as well as all the combos associated with each dimension. These combos and the challenge name are used as a key to search the hash table. If there is a match, the sliding “window” is returned. Then, the model computes the skill in this dimension. The player’s skill measurement in each pair is calculated through a weighted arithmetic mean, the skill \(S\) for a given pair being computed by the following expression:

\[
S = \frac{\sum_{i=1}^{A} w_i \cdot A_i}{\sum_{j=1}^{A} w_j}
\]  

(4.2)
where $A_i$ represents each attempt, rated by a numerical value between 0 (total failure) and 1 (total success): $[0, 0.5 \cup 0.5, 1]$. Values between 0 and 0.5 represent negative states such as, for example, totally unsuccessful or near-miss attempts. Values in between 0.5 and 1 represent successful attempts like a clutch success or a fully successfully attempt. Each attempt in this “window” of most recent attempts contributes with a different weight $w_i$ to the skill assessment of the gameplay element. In our case study, the chosen weight distribution was linear (specifically: $w_1 = 1, w_2 = 2, \ldots, w_{10} = 10$) to emphasize that the more recent attempts are more relevant when evaluating player skill than previous attempts.

4.6 Reward System

Rewards were added to pique the player’s curiosity through gameplay diversity in an otherwise very simple and repetitive game while encouraging the player to get better as the game progressively becomes more difficult.

![Figure 4.3](image1.png)

(a) If a player is more skilled at ‘double jump’ than ‘dash’, he/she may choose to swap the challenges and engage the one originally on top using the ‘double jump’ mechanics, when the challenge was tuned to promote the ‘dash’ mechanics.

![Figure 4.3](image2.png)

(b) By using rewards, we disclose more clearly what is happening under the hood and promote player engagement with the adaptation systems, while still giving the player the total freedom to choose how to best overcome the challenge.

Figure 4.3: The two possible paths one can take to overcome the Dash-box challenge

Rewards were also introduced to mitigate an issue that appeared during our preliminary testing. Figure 4.3 gives an example of this issue and illustrates how the introduction of rewards helped. The scenario on the left (without rewards) depicts what would typically happen when a player had to choose which challenge to overcome based on the two proposed challenges previously evolved to create meaningful choice while providing with an overall equally challenging obstacle (e.g. in this case, the volume of both obstacles would be roughly the same). As players would get better at using certain mechanics over others, they would use these mechanics consistently but would not push themselves to get better at them, typically choosing the path demanding less effort. In the example of Figure 4.3, and assuming the player had developed a better skill at ‘double jump’ than ‘dash’, the typical course of action would be to switch between the two challenges then use ‘double jump’ to overcome the challenge originally presented on top, when that particular challenge was evolved to encourage the player to get better at ‘dash’ mechanics.

While our model is robust to such an effect, this issue does not encourage the player to improve on the skills she had developed and maintain her engagement with the game. As we believe both the
game and the player should participate actively in the adaptation process, we introduced rewards to mitigate this effect and better disclose the state of the adaptation process. To motivate the player to get better at the game, rewards were introduced to highlight the path related to the mechanics being evolved by our adaptation process. While not preventing the player from choosing any other alternative way of overcoming an obstacle, it would make the adaptation more visible through the creation of a (very slightly) higher risk, higher reward path. In the example of Figure 4.3, rewards were placed to disclose the ‘path of the dash’ and the ‘path of the jump’ that influenced the evolution of both challenges. When a reward is present, it is present in both the top and bottom challenges, to ensure equally rewarding choices in terms of the impact on the player score (it is after all, a high-score single player game). An important note is that the values for the rewards provided are quite low when compared to the reward of surviving for a longer period of time in the game.

Again, this is only an issue of communicating more effectively what is happening under the hood to the player to encourage engagement with the game systems: the player is always free to choose how to combine the four available mechanics to overcome each obstacle. And if the player chooses a different path, our model will seamlessly adapt to it.

4.7 Summary

In this chapter, we described in detail our model. We explored how player skill progression can be used in PCG to create games better adapted to each player, and created a model for online content adaptation using player skill progression as a core feature.

We started providing a general overview of the model and how it operates. Then, we detailed how challenges require at least two different paths to the player to overcome. The game allows players to chain mechanics, we detailed this concept of combos and how they are used in the model. One of the crucial aspects of our model is to adapt the game to the player’s skill progression. However, to do this, the model needs to know how well the player is doing over time at using the different game mechanics to overcome the different challenges proposed by the game. To that end, we proposed that player performance could be measured based on the recent performance at using a certain combo of mechanics to overcome a certain challenge, i.e. associating a success rating with all pairs (challenge, mechanics) used by the player. By measuring the player performance in each one of these dimensions, it is possible to infer which dimension the player is currently better at when confronting a certain challenge and use this information to progressively increase the difficulty in a manner that will encourage the player to improve at the game and promote engagement with its gameplay systems. Finally, we described why the reward system was added and how it can motivate player to get better.
Chapter 5

Evaluation

In this chapter, we will describe the evaluation process used to validate our model. We start by briefly detailing our preliminary evaluations, conducted during the development stages. Secondly, and most importantly, we explain in detail the main procedure used in the evaluation of the final version of the proposed solution, including the play tests, the questionnaire and what has been logged and why. It is worth mentioning that our model never acts randomly, not even in the beginning, because there is always some data gathered previously from the tutorial play session that players are required to play at least one time.

5.1 Experimental Approach

We hypothesized that difficulty cannot be viewed as a global setting affecting the whole game, difficulty should be divided into various dimensions since each player's proficiency varies in each of these dimensions, adapting to each player differently; the player expresses preference for certain dimensions in detriment of others, considering the context of a game with a progressive increasing difficulty. Initially, in order to verify our hypothesis, we considered to use the Game Experience Questionnaire [28] as a way to evaluate our approach, but instead, the element which will help us to evaluate our approach is embedded in the game itself. Which is a much more direct way (since for instance, there isn’t a break in the flow like using questionnaires after the play sessions) avoiding interviewing players extensively about their experience.

The validation process used to test our model consists in providing the player with the option to choose between two variations of a challenge each time the player has to overcome this challenge in the game (see figure 5.1). These two variations, the plus mode of the challenge and the minus mode of the challenge, represent two different evolutions of the same original challenge. The plus mode is the result of evolving the challenge according to the dimension of play the player is the most skilled at that time, while the minus mode is the exact opposite. Each time the minus mode challenge evolves, it does so in the dimension of play the player is the least skilled. The minus mode was added into the game with the purpose of validating the viability of our model comparing with the Plus mode.

Once evolved, one of the modes is spawned on the top lane while the other is spawned on the bottom lane. The player is then provided with the option to choose between the two versions of the challenge, without being explicitly told which variation is the plus mode and which variation is the minus mode. Additionally, the player has no prior knowledge of the existence of this process. The player is able to choose the top lane challenge through the switch mechanics or simply wait for the bottom lane challenge to reach her.
The number of Plus and Minus mode challenges are balanced between each lane, i.e., the bottom and top lane. We ensure that, in a window of six challenges, the plus mode of the challenge will be placed thrice on the top lane and thrice on the bottom lane, although how this happens in the window is totally random. This positioning is to make sure the player does not need to intercalate the usage of the switch every challenge in case they choose to intuitively attempt every plus mode challenge.

With the purpose of verifying our model, we will analyze if players preferred mostly the plus mode (+) of the challenge over the minus mode (−) of the challenge, and if there was in fact a preference for switching to the plus mode challenge over switching to the minus mode challenges, supporting our hypothesis that players express their preference for certain dimensions in detriment of others and that games should take this into account when modeling difficulty progression.

5.2 Preliminary evaluations

The preliminary evaluations' main objective was to track and solve usability issues that our game could have. Preliminary playtests were performed in early stages of the model and game development. The preliminary tests were performed in small groups (4 to 6 people) with multiple iterations throughout 2 to 3 months. The preliminary evaluations did not follow a formal procedure. The consequent changes emerged from informal discussions and interactions with the game.

5.2.1 Changes

Multiple iterations were performed in order to better validate our game’s usability as well as the model. Many flaws were detected and solved that could potentially be problematic perceiving the game correctly. Some of the flaws detected and consequent changes:
- Collision Issue: The issue was that players could not sometimes understand when and how did a collision happen between the player character and the challenges. In order to further display the collision, a visual effect around the player character was added that represents the collision. This visual effect also provides the player a small interval of time of challenge immunity i.e. invincibility for a small amount of time while the visual effect is turned on.
- Too fast to learn: We detected some players could not learn the mechanics properly in time and would lose too fast. To solve this the tutorial was created. The tutorial allowed players to know which keys to press in each challenge as well as it would provide enough time to players to react.
- Revert the Switch: Some players would, in some occasions, press the Switch button (to switch the challenges) unintentionally. To solve this, we have added a revert switch functionality. By pressing...
the switch button again while the switch was still going, the switch would revert back to its original position, reverting the switch action.

– Lives: Some players could not perceive that the white squares at the top corner represented the current lives. To fix this, an animation was added so that when the player collided with a challenge, the lost live would flash in and out sometimes before disappear totally.

### 5.3 Final evaluation

Our final evaluation had a participation of 30 players, where each player was asked to perform two tasks: answer a short questionnaire (for demographic basic information such as age and gender) followed by a play test of our game. The questionnaire is available in appendix A. This questionnaire contains exclusively demographic related questions, in order to better know the test population.

### 5.3.1 Web Application

The full test was performed autonomously by all participants using a PC or Mac computer with an Internet connection. The web application provided all the information required for any player to execute the test. The web application provided a simple introduction, and the test instructions as well as the download link for both platforms (windows and mac) and the questionnaire link. This website can be viewed in appendix C. In order to link the questionnaire data with the play session logs data, a unique ID system was created. Any player that would execute our test would need to first create an unique ID. To generate this ID, the player had to press the “create ID” button located in our web application. Then, the player had to introduce this ID, in the questionnaire and in the game. The game would only start if a correct ID was introduced.

### Implementation

This web application was implemented in React and Node.js. The web server is Nginx. The ID generation was done using the uniqid package from npm. Each time a player reaches game over by losing all lives, the logs are sent to our server through a simple API using JSON data. The JSON data is then stored in a MongoDB hosted in mLab. The API was created using Express, a framework for Node.js. The validation of the user was done through tokens. We used the tokens provided by jsonwebtoken package from npm. Each time the ID was inserted in the game, a token was generated to that user to be used in the API to send the logs in JSON.

### 5.3.2 Play session

The playtest procedure for every participant was to: play the tutorial at least once and then play the normal game, although the players could play the tutorial and the game multiple times if they so desired. Therefore the logs, for each play session, depend on how much time each player spent playing and how many times. The positioning of the plus challenges and minus challenges (top lane or bottom lane) is done in a balanced manner so there is no bias concerning the choice of the player as explained previously in the Experimental Approach section.
5.3.3 Logs

The logged data was exported to a CSV file and imported into IBM SPSS for analysis for a quicker way to visualize the data. The logged data includes the sequence of challenges, the position of the different modes of each challenge, which mode the player attempted to overcome and if a switch command was executed before the attempt.

![Table of log data](image)

Figure 5.2: Part of a log from a play session.

Part of a log is depicted in figure 5.2. The first row denotes the session number and if it was tutorial or the normal game, in this case it is the player’s second game. The rest of the log is structured in a simple manner: the first column denotes the challenges, the second column represents the bottom lane challenge mode, the third column represents the top lane challenge mode, the fourth column represents the mode chosen by the player and finally, the last column represents whenever a switch was used or not. In this example, the player chose the plus (+) mode challenges 5 times in 6 challenges and consequently 1 time the minus (−) mode. The player also used the switch two times (fourth and last row) to attempt the plus (+) mode challenges.
Chapter 6

Results

This chapter presents the analysis of the data collected in the final evaluation regarding the questionnaires and the play sessions. We start by describing our population sample then, provide a detailed statistical analysis of the player logs as a way to evaluate it and to conclude whether or not our progression model is valid and viable. The statistical analysis was processed using the IBM SPSS Software version 24.

6.1 Sample

The final evaluation was carried out with the participation of 30 persons with ages between 16 and 30 years old (M = 22.97, SD = 3.77, 7 female). The figure 6.1a shows that participants were split evenly between casual players and dedicated players, and 16.7% do not play games. The figure 6.1b displays that most participants were familiar with endless runner games (60%), 23.3% were not familiar with endless runner games although familiar with games in general. The demographic information was acquired using the questionnaire in appendix A. No statistically significant effect was found when comparing the different demographic groups.

6.2 Play Time

As explained in the Evaluation chapter, players had to play at least one time the tutorial before playing the normal game. For the tutorial, the time users played, ranged between 2.25 and 16.53 minutes (M = 5.50, SD = 3.66). For the normal game, the time played ranged between 2.05 and 20.15 minutes (M = 6.83, SD = 4.86).
6.3 Normality Test

Before beginning our analysis, we need to know whether the ratios that we will use follow a normal distribution or not. In order to know which approach we should use, i.e, parametric (t-student) or non-parametric tests, a Shapiro-Wilk test was conducted.

The result of the Shapiro-Wilk test showed that most of the calculated ratios do not follow a normal distribution, as such, we will use non-parametric tests to analyze our data. The tests are shown in the table 6.1.

![Graph of video-game frequency]

![Graph of user familiarity with endless runner games]

Figure 6.1: Participants Information.

<table>
<thead>
<tr>
<th>Ratio</th>
<th>Statistic</th>
<th>df</th>
<th>Sig</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plus_Ratio_Total</td>
<td>.895</td>
<td>30</td>
<td>.006</td>
</tr>
<tr>
<td>Minus_Ratio_Total</td>
<td>.895</td>
<td>30</td>
<td>.006</td>
</tr>
<tr>
<td>Switch2plus_Ratio</td>
<td>.754</td>
<td>30</td>
<td>.000</td>
</tr>
<tr>
<td>Switch2minus_Ratio</td>
<td>.861</td>
<td>30</td>
<td>.001</td>
</tr>
<tr>
<td>Plus_Ratio_1</td>
<td>.963</td>
<td>30</td>
<td>.358</td>
</tr>
<tr>
<td>Plus_Ratio_2</td>
<td>.958</td>
<td>30</td>
<td>.277</td>
</tr>
<tr>
<td>Plus_Ratio_3</td>
<td>.931</td>
<td>30</td>
<td>.053</td>
</tr>
<tr>
<td>Minus_Ratio_1</td>
<td>.963</td>
<td>30</td>
<td>.358</td>
</tr>
<tr>
<td>Minus_Ratio_2</td>
<td>.958</td>
<td>30</td>
<td>.277</td>
</tr>
<tr>
<td>Minus_Ratio_3</td>
<td>.931</td>
<td>30</td>
<td>.053</td>
</tr>
</tbody>
</table>

Table 6.1: Shapiro-Wilk Normality Test
6.4 Overall Player Preference Test

In order to know which mode (Plus or Minus) the players preferred, we conducted a Wilcoxon signed-rank Test between two ratios: $Plus\_Ratio\_Total$ and $Minus\_Ratio\_Total$. These ratios were calculated using the following formulas:

\[
Plus\_Ratio\_Total = \frac{Total\_Plus\_N}{Total\_Plus\_N + Total\_Minus\_N}
\]  
\[6.1\]

\[
Minus\_Ratio\_Total = \frac{Total\_Minus\_N}{Total\_Plus\_N + Total\_Minus\_N}
\]  
\[6.2\]

Where $Total\_Plus\_N$ refers to the total number of times that a player attempted the plus mode challenges and the $Total\_Minus\_N$ refers to the total number of times a player chose to do the minus mode challenges. These values include the result of switching, e.g., if a player switched a minus challenge for a plus challenge, $Total\_Plus\_N$ value is increased by one.

**Results**

The $Plus\_Ratio\_Total$ presents a median of 0.59 (0.46 to 0.88, IQR = 0.13) whereas the $Minus\_Ratio\_Total$ the median is 0.41 (0.13 to 0.54, IQR = 0.13). The Wilcoxon signed-rank test showed that there is a statistically significant change when comparing the player preference between $Plus\_Ratio\_Total$ and $Minus\_Ratio\_Total$ ($Z = -4.742, p < 0.01$), with a large effect size ($r=0.61$).

This test suggests that players, in fact, have a statistically significant preference for plus mode challenges than minus mode challenges throughout the play sessions.

6.5 Switch Usage Test

As the presentation of the challenges by the game was balanced between plus and minus for each lane (top and bottom), the idea of this test is to analyze if the players opted to switch more into plus mode challenges (when minus mode challenges were presented in the bottom lane) than to switch into minus mode challenges (when plus mode challenges were presented in the bottom lane). A Wilcoxon signed-rank was conducted between two ratios: $Switch2plus\_Ratio$ and $Switch2minus\_Ratio$. These ratios were calculated using the following formulas:

\[
Switch2plus\_Ratio = \frac{Switch2plus\_Total\_N}{Minus\_Total\_N + Switch2plus\_Total\_N}
\]  
\[6.3\]

\[
Switch2minus\_Ratio = \frac{Switch2minus\_Total\_N}{Plus\_Total\_N + Switch2minus\_Total\_N}
\]  
\[6.4\]

$Switch2plus\_Total\_N$ refers to the total number of times the player switched from a minus mode challenge to a plus mode challenge in the game. The $Minus\_Total\_N$ variable refers to the total number of times a player did not switch when a minus was presented (and therefore chose the minus mode without switching) . The sum in the denominator of the first equation simply refers to the total number of times a minus mode challenge was presented in the bottom lane (i.e. the default choice in case the player did not switch, facing a minus challenge) to the player. For the second equation, the denominator refers to the total number of times a plus mode challenge was presented in the bottom lane to the player.
Results

The Switch2plus_Ratio presents a median of 0.37 (0.11 to 0.80, IQR = 0.30) whereas the Switch2minus_Ratio the median is 0.07 (0.00 to 0.31, IQR = 0.17). The Wilcoxon signed-rank test showed that there is a statistically significant change when comparing the player preference in switching into the plus mode or the minus mode, i.e., Switch2plus_Ratio and Switch2minus_Ratio, $Z = -4.762$, $p < 0.01$, since the $p$ value is inferior to 0.05, with a large effect size ($r=.61$).

The interpretation of this analysis is that when players opted to use the switch, they switched significantly more into plus mode challenges than minus mode challenges.

6.6 Segmented Player Preference Test

For this test, instead of looking to the total values of each play session, we divided the play sessions equally into three segments. To explain how this was done, imagine a play session where the player overcame 45 challenges, in this case we would define the initial segment of the play session from the 1st to the 15th challenge, the middle segment from the 16th to 30th and finally the final segment would be defined from the 31st to 45th. The purpose of dividing the play sessions into segments is to identify if there are actual differences in preference between each segment, instead of only analyzing the play session as a whole (like in section 6.4). A Wilcoxon test was conducted for each Plus and Minus ratios corresponding to each segment.

These ratios were calculated similarly to the equations 6.1 and 6.2 but instead of using the total ratios values, it was used the ratio values for each of the three equal segments.

Results

For the initial segment (Plus_Ratio_1 and Minus_Ratio_1), the medians were respectively 0.60 (0.25 to 1.00, IQR = 0.17) (plus ratio) and 0.40 (0.00 to 0.75, IQR = 0.17) (minus ratio). The Wilcoxon signed-rank test revealed a statistically significant change when comparing the player preference between Plus_Ratio_1 and Minus_Ratio_1, $Z = -2.842$, $p = 0.004$, with a medium effect size ($r = 0.36$).

For the middle segment (Plus_Ratio_2 and Minus_Ratio_2), the medians were respectively 0.62 (0.00 to 1.00, IQR = 0.24) (plus ratio) and 0.38 (0.00 to 1.00, IQR = 0.24) (minus ratio). The Wilcoxon signed-rank test revealed a statistically significant change when comparing the player preference between Plus_Ratio_2 and Minus_Ratio_2, $Z = -3.123$, $p = 0.002$, with a medium effect size ($r = 0.40$).

For the final segment (Plus_Ratio_3 and Minus_Ratio_3), the medians were respectively 0.62 (0.33 to 1.00, IQR = 0.25) (plus ratio) and 0.38 (0.00 to 0.67, IQR = 0.25) (minus ratio). The Wilcoxon signed-rank test revealed a statistically significant change when comparing the player preference between Plus_Ratio_3 and Minus_Ratio_3, $Z = -3.776$, $p < 0.01$, with a relatively large effect size ($r = 0.49$). The test results can be seen in the table 6.2.

This test demonstrates that players prefer the plus mode when comparing with the minus mode from the beginning and continue to do so, over time. The difference between plus mode challenges and minus mode challenges become even greater over time. The plus mode becomes even more adapted to their skill and consequently the minus mode even less adapted to their skill.
6.7 Segmented Wilcoxon Signed-Ranks Test Usage Test

To do this test, we also segmented the play sessions into three equal segments like the section 6.6. This test is to analyze how much players switched into what mode and in which segment. A Wilcoxon signed-rank test for Switch2plus_Ratio and Switch2minus_Ratio were conducted for each segment, similarly to section 6.6. The ratios were calculated similarly to section 6.5 with the only difference being the usage of the correspondent variables for each segment instead of using the total variables.

Results

For the initial segment, the medians were respectively 0.33 (0.00 to 1.00, IQR = 0.36) (Switch2plus_Ratio_1) and 0.00 (0.00 to 1.00, IQR = 0.25) (Switch2minus_Ratio_1). The Wilcoxon signed-rank test revealed a statistically significant change when comparing the player preference in switching into the plus mode or the minus mode for the initial segment, $Z = -2.892$, $p = 0.004$, with a medium effect size ($r = 0.37$).

For the middle segment, the medians were respectively 0.20 (0.00 to 1.00, IQR = 0.50) (Switch2plus_Ratio_2) and 0.00 (0.00 to 0.60, IQR = 0.00) (Switch2minus_Ratio_2). The Wilcoxon signed-rank test revealed a statistically significant change when comparing the player preference in switching into the plus mode or the minus mode for the middle segment, $Z = -3.346$, $p = 0.001$, with a medium effect size ($r = 0.43$). In this segment of the play session, players displayed a tendency for not using the switch as much as the initial segment. Although there was an increase comparing the Switch2plus and Switch2minus ratios with the initial segment.

For the final segment, the medians were respectively 0.45 (0.00 to 1.00, IQR = 0.52) (Switch2plus_Ratio_3) and 0.00 (0.00 to 0.50, IQR = 0.13) (Switch2minus_Ratio_3). The Wilcoxon signed-rank test revealed a statistically significant change when comparing the player preference in switching into the plus mode or the minus mode for the final segment, $Z = -4.092$, $p < 0.01$, with a large effect size ($r = 0.53$). The test results can be seen in the table 6.3.

These tests reveal that players opted to switch into the plus mode significantly more than minus mode over time in a consistent manner.

<table>
<thead>
<tr>
<th>Wilcoxon Signed Ranks Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switch2plus_Ratio_1</td>
</tr>
<tr>
<td>Switch2minus_Ratio_1</td>
</tr>
<tr>
<td>$Z$</td>
</tr>
<tr>
<td>Asymp. Sig. (2-tailed)</td>
</tr>
</tbody>
</table>

Table 6.2: Wilcoxon Signed-Ranks Tests of the three segments (Plus and Minus Ratios).

<table>
<thead>
<tr>
<th>Wilcoxon Signed Ranks Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switch2plus_Ratio_1</td>
</tr>
<tr>
<td>Switch2minus_Ratio_1</td>
</tr>
<tr>
<td>$Z$</td>
</tr>
</tbody>
</table>

Table 6.3: Wilcoxon Signed-Ranks Tests of the three segments (Switch2plus and Switch2minus Ratios).
6.8 Comparison of Preference Between Segments

For this comparison we conducted another statistical test, the Friedman test. Friedman test is another non-parametric test which is used to detect differences in treatments across multiple test attempts. We examined the three different segments for each individual ratio: the Plus, Minus, Switch2plus and Switch2minus.

**Plus Mode Preference Comparison**

The Friedman test showed that there was no statistically significant difference ($p > 0.05$) between each segment (the initial, middle and final segments) regarding the preference between plus and minus modes, $\chi^2(2) = 1.264$, $p = 0.531$. Although the values of the mean ranks increase gradually each segment: Plus\_Ratio\_1’s Mean Rank = 1.85, Plus\_Ratio\_2’s Mean Rank = 2.03, Plus\_Ratio\_3’s Mean Rank = 2.12, the test did not found that this increase was statistically significant.

**Minus Mode Preference Comparison**

The Friedman test showed that there was no statistically significant difference ($p > 0.05$) between each segment regarding the preference between plus and minus modes, $\chi^2(2) = 1.264$, $p = 0.531$. Although the values of the mean ranks decrease gradually each segment: Minus\_Ratio\_1’s Mean Rank = 2.15, Minus\_Ratio\_2’s Mean Rank = 1.97, Minus\_Ratio\_3’s Mean Rank = 1.88, the test did not found that this decrease was statistically significant. This was expected since theses minus ratios complements the plus ratios.

**Switching into Plus Mode Preference Comparison Test**

The Friedman test showed that there was in fact a statistically significant difference ($p < 0.05$) between each segment in preference of switching into plus mode challenges, $\chi^2(2) = 8.072$, $p = 0.018$.

In order to find between which segments this difference actually occurs we conducted post hoc analysis with Wilcoxon signed-rank tests between each segment with a Bonferroni correction applied, resulting in a significance level set at $p < 0.017$.

The results showed that there were no statistically significant differences between the initial and middle segments ($Z = -0.216$ $p = 0.829$) or initial and final segments ($Z = -1.172$ $p = 0.085$). However, there was a statistically significant increase in preference for plus mode challenges from the middle segment to the final segment ($Z = -3.044$ $p = 0.002$). This analysis suggests that in the initial segment there is an exploratory phase where players experiment different mechanics. In the middle and final segments the analysis conveys that players still continues to use the preferred mechanic to overcome a challenge even when the difficulty is reaching its peak while there is an option of using another mechanic to overcome the challenge in another dimension.

**Switching into Minus Mode Preference Comparison Test**

The Friedman test showed that there was a statistically significant difference ($p < 0.05$) between each segment in preference of switching into minus mode challenges, $\chi^2(2) = 6.633$, $p = 0.036$.

In order to find between which segments this difference actually occurs we conducted post hoc analysis with Wilcoxon signed-rank tests between each segment with a Bonferroni correction applied, resulting in a significance level set at $p < 0.017$. 
The results showed that there were no statistically significant differences in preference of switching into minus mode challenges between the initial and middle segments (Z = -2.262 p = 0.024) or initial and final segments (Z = -1.068 p = 0.286) or even between the middle and final segments (Z = -1.201 p = 0.230).

6.9 Discussion

The final evaluation was concerned to evaluate the correctness of our implemented Progression Model and consequently our hypothesis. We hypothesized that: difficulty cannot be viewed as a global setting affecting the whole game, difficulty should be divided into various dimensions since each player’s proficiency varies in each of these dimensions, adapting to each player differently; players express preference for certain dimensions in detriment of others, considering the context of a game with a progressive increasing difficulty.

Our approach to test our model consisted in implementing within the game a validation process where we could extract information about the player preference. This validation process provided the player with the ability to choose between two challenges each time: one of the challenges was generated with our model, which we denominated the plus mode, and the other challenge was generated the opposite way our model was, the minus mode. Through the acquired play logs from the play sessions, we obtained player data relative to the choice between these two challenges. This way we managed to avoid the traditional experience questionnaires, where players would have to answer extensive questionnaires about their game experience.

The analysis of the obtained data was accomplished using several non-parametric statistical tests regarding the player choice between these two modes. The results of our statistical analysis showed there was a significant difference when comparing the preference between players choosing the plus and minus modes. Players chose significantly and consistently more, the plus mode over the minus mode over time. The analysis suggests that the initial segment is an exploratory phase where players experiment different mechanics, even though there is still significant preference for plus mode over minus mode, it is not as significant as in later segments. Players express even more preference for plus mode over minus mode in later segments, revealing an increasing preference for plus mode over time which consequently reveals a decreasing preference for minus mode.

We also verified that there is a statistically significant difference in switching into plus mode challenges when comparing the middle and final segments. Our interpretation of this result is that players continue to use the preferred mechanic to overcome a challenge, even when the difficulty is reaching its peak while there is an option of using another mechanic to overcome the challenge in another dimension. This supports our hypothesis that players express preference for certain dimensions in detriment of others in the context of a game with a progressive increasing difficulty, and, as such, difficulty should not be modelled as a global setting in games where player adaptation is key.
Chapter 7

Conclusions

In this chapter, we summarize our work describing our game and the progression model, as well as, drawing out meaningful conclusions as to whether the developed progression model demonstrated to be a better game experience to the players. We also mention several possibilities that our work can progress into, for future work.

7.1 Summary

In this work, we explored how player skill progression could be used in PCG to create games better adapted to each player, and proposed a model for online content adaptation using player skill progression as a core feature.

To adapt the game to the player's skill progression, the model needs to know how well the player is doing over time at using the different game mechanics to overcome the different challenges proposed by the game. Thus, the model starts by measuring the player skill. Instead of measuring the overall player skill, we started by defining skill in terms of mechanics (or combos) and corresponding challenge. Each time a player overcomes or fails a specific challenge the model registers the combo used to overcome the challenge, the particular challenge, as well as the player's success. This information is saved as pairs \(<\text{challenge},\text{mechanics}>\). To inform the selection of the dimension to evolve, our model maintains the player skill value for all pairs (mechanics and corresponding challenge) used by the player during the game. In each pair we keep a window of 10 attempts which we can then calculate the skill by using a weighted linear mean of the attempts. Every challenge in the game have at least two dimensions. These dimensions are associated with the how they are surpassed, i.e. in each challenge there is at least two possible different paths (using different combos) that the player must choose to be able to overcome it. These combos are associated to different dimensions of the challenge, in order to search the data using the challenge name and each one of the combos. For every match, the model calculates the skill associated with overcoming the challenge with this combo based on the most recent attempts, then compares these calculations to find the dimension with the best skill. After this step, the model knows exactly which pair has the best skill and also which dimension it is associated to. Therefore, the model is now able to increase the challenge’s difficulty in this particular dimension.

After defining our model, our approach to test our model consisted in implementing within the game, in a complementary manner, a validation process where we could extract information about the player preference, instead of interviewing players extensively about their experience, for instance, relative to two different versions of the game with different model settings. Providing the player with the ability to choose between multiple challenges which one the player prefers to attempt is a much more automatic
process to assess alternatives in a quantitative perspective. In the game, the player at every attempt has to choose between two challenges, these two challenges have two different evolution forms: the Plus and Minus modes.

The Plus mode represents the model’s algorithm behind the evolution of challenges based on the player skill. The plus mode’s main objective is to, on every challenge spawn, increase difficulty based on which way the player was more skilled i.e. the highest calculated skill for each pair. Whereas the minus mode, behaves exactly the opposite way of the plus mode i.e. instead of increasing the difficulty, where the player is supposedly more skilled, it increase the difficulty in where the player is worst. The minus mode was added solely as a way to validate our model (plus mode). Both modes difficulties are increased equally meaning that the global difficulty of the challenges are identical after any adaptation.

Our developed progression model was applied to our custom made game “Go, Go Hexahedron!”. “Go, Go Hexahedron!” is an endless running side-scrolling platform game with a minimalist aspect featuring a white cube as the player controllable character. In “Go, Go Hexahedron!” the player character, white cube, is continuously “running” towards the right of the screen and its objective is to obtain the most points possible while avoiding every obstacle in the way. These points are obtained staying alive (the longer, the more points the player can get), and by catching the different reward spheres. One of the core feature of the game is the ability provided to the player to choose in each confront from two different challenges. The chosen challenge reflects the one that it is more appealing to the player’s interests. Both challenges have the same base form but evolve in different ways which is directly related to the players’ actions. The foundation of our evaluation is this exact element of choosing between challenges, where we analyzed each choice from every player in order to understand if our underlying progression model is indeed more attractive and adapted to each player.

In the final evaluation, we had a participation of 30 players, where each player was required to answer a short questionnaire, followed by a play test of our game. Our analysis focused on the logs obtained from the play sessions of each player. In the logs we obtained information concerning the player choices. We conducted various non-parametric tests to our data from the logs with the aim of solidifying our model validation. The results of our statistical analysis showed there was a significant difference when comparing the preference between players choosing the plus and minus modes. Players chose significantly and consistently more, the plus mode over the minus mode over time. The analysis suggests that the initial segment is an exploratory phase where players experiment different mechanics, even though there is still significant preference for plus mode over minus mode, it is not as significant as in later segment. Players express even more preference for plus mode over minus mode in later segments, revealing an increasing preference for plus mode over time. We also verified that there is a statistically significant difference in switching into plus mode challenges when comparing the middle and final segments. These results suggest there is a significant pattern in player choice, with players consistently preferring overcoming the challenges through the use of the skills they are more proficient at, even when the difficulty is reaching its peak and they have the option of using another set of mechanics to overcome such challenges. These results support our hypothesis that difficulty should not be modelled as a global setting in games as well as that players express preference for certain dimensions in detriment of others, considering the context of a game with a progressive increasing difficulty. Knowing the specific preferences of each player based on the progression of their skill at overcoming the various dimensions of play offered by the game could provide invaluable insight for the generation of content adequately adapted to each player.

Finally, we submitted the extended abstract of this work as a full paper which was posteriorly accepted to appear in the International Conference on the Foundations of Digital Games (FDG). The FDG conference follows the peer-reviewed papers process. This scientific paper explains this work in a more concise and summed manner focusing on the multi dimensional player skill progression model and the
embedded evaluation process.

7.2 Future Work

Throughout the development of our game and model until the final evaluation, we found several alternatives or ideas to improve certain aspects concerning the developed progression model. These were the most relevant ones we kept for future work:

The creation of more challenges with more dimensions could prove as an interesting starting point to provide players with more diversity within challenges. The current challenges have two different possible paths, but our model was made to support any number of paths. Creating more challenges with more paths could improve the player's experience. Another possible way to make the game even more diverse is to create challenges with different growths, i.e. instead of making the model increase the size of some objects in the challenge, it could potentially be used to increase the number of slicing blades or lasers in new challenges.

Another interesting change could be to change the detection mechanism of the attempts into a continuous form (instead of the current discrete one). At the moment the model registers in each attempt if the player successfully overcome the challenge or not. It could be beneficial to add a more continuous approach, i.e., making the model to detect how well a player did in each attempt for each pair \(<\text{challenge, mechanics}>\). Even though, the current use of discrete detection has still given positive results on our analysis, this way the model could be provided with more accurate information about each attempt and thus it could possibly improve the outcome of the results.

An improvement for the challenges adaptation process could be based on the previous idea: by using the continuous approach we could tweak how much growth is applied to each way in each dimension based. Instead of using a logarithmic growth, we could decide how much each challenge evolve based on how the player was rated in each pair \(<\text{challenge, mechanics}>\).

A more ambitious change could be to adapt the model to operate in another video game genre. The progression model implementation itself is very modular and thus could be used to increase the difficulty of another game. Although it cannot be any game, the game requires some essential features such as: it needs to have some sort of different unique challenges that the player needs to surpass, as well as, these challenges need a way to evolve, so the model could be applied to these challenges.
Bibliography


Appendix A

Questionnaire

Hello, I'm developing a game for my Master's Thesis at IST and need some help to test it. Thank you for your time.

* Required

What is your gender? *
- Female
- Male
- Other

What is your age? *
Your answer

How often do you play video-games? *
- I don't play video-games
- I play video-games occasionally when the opportunity presents itself
- I make some time in my schedule to play video-games.

Are you familiar with endless runner games? *
- I don't play video-games
- I play games but I never played an endless runner game
- I am familiar with endless runner games and play them occasionally

ID? (copy paste from website) *
Your answer

Figure A.1: Questionnaire presented to the players before playing the game.
Appendix B

Case Study Interface appendix

The game’s main menu, as shown in figure B.1, displays four different buttons:

- The play button: This button allows the player to view the next screen where he/she can choose between the tutorial or the normal game (until the player plays at least one time the tutorial mode the normal button will maintain unaccessible).
- The scores button: by pressing this button, the player can check his or her own achieved scores and compare them. The scoreboard is illustrated in figure B.2.
– The how to play button: Pressing this button, the player will be taken to the how to play menu (displayed in figure B.3) where the player may read varied information regarding the game, such as, general knowledge of the game, for example, the rules and the objectives, (displayed in figure B.4), the game controls (displayed in figures B.5 and B.6) and the description of the game’s challenges (displayed in figure 3.5).

– The exit button, where the player may press if he wishes to quit the game.
Figure B.4: General Info Screen

The game objective is to reach the farthest possible by overcoming the presented challenges using a certain mechanic or a combination of mechanics. To facilitate the learning process there is a tutorial mode which provides an easier, yet simple way to learn how to play.

When the tutorial mode is activated the approaching challenges cause the game to slow down allowing the player to decide between the two presented challenges (between the top challenge and the bottom challenge). This allows the player to choose more precisely the desired challenge without feeling rushed to decide. To further assist the player, it’s also possible to revert the switch while it is still going by pressing the switch button again.

After the player chooses between the challenges, the game’s speed will return to normality and small tooltips containing buttons will show up. Pressing these buttons will allow the player to overcome the current challenge.

Rewards are green, blue and yellow spheres with a plus sign. A visual representation of this reward can be seen here:

When the player catches a reward it’s awarded with 5 (green reward), 10 (blue reward) or 15 (yellow reward) additional points.

RETURN

Figure B.5: Controls page using a keyboard. By pressing any of the shown keys, more information will be provided for the respective pressed key. In this case, the upper arrow key was pressed, providing information concerning the jump mechanic.
Figure B.6: Controls page using a controller. By pressing any of the shown keys, more information will be provided for the respective pressed key. In this case, the upper arrow key was pressed, providing information concerning the jump mechanic.
Appendix C

Web Application

![Website](image)

Figure C.1: Website presented to the players before playing the game.
Appendix D

Tutorial mode

The tutorial begins by presenting the six different challenges, one by one, accompanied with mechanical tool tips to further help the player. In this phase, only the bottom lane is used, thus these six challenges will only appear in the bottom lane challenge, which therefore, the player is unable to use the switch button (i.e. there isn’t a top lane challenge to switch to). This phase is shown in figure D.1. Needless to say, since there aren’t two challenges the player can swap between, the slow motion is not applied to these six challenges. These challenges will always be presented in the same order and in their initial difficulty since this part is independent of the player’s actions. This phase was designed to introduce players to every challenge in their easiest form, but also to gather some initial player data regarding their skill in each challenge to initialize our progression model. Because of this, the game requires the player to play at least one time the tutorial. In this phase the model is not evolving any challenge and therefore rewards do not appear.

![Figure D.1: A screenshot of the first phase of the tutorial.](image)

After this phase, the challenges are presented as in the normal game, i.e. there is also a top lane challenge and bottom lane challenge with rewards similarly to the normal game. But as explained, a slow motion is applied until the player chooses which challenge to face displaying a tool tip as well. After this choice the tutorial display another tool tip regarding the buttons the player may press to overcome the current challenge. This tool tip is also dependent on the player’s input device (game pad or keyboard).
An example of this tool tip can be found in figure D.2.

The tutorial mode solves the problem where players might be overwhelmed by the time pressure and thus might not be able to switch. The tutorial is a way to obtain more precise data from the players’ choices, without having time as a factor in the player’s decision making.