Game Content Placement through Progression Modeling

Bruno Alexandre Ferreira de Almeida

Thesis to obtain the Master of Science Degree in
Information Systems and Computer Engineering

Supervisor: Prof. Carlos António Roque Martinho

Examination Committee

Chairperson: Prof. Daniel Jorge Viegas Gonçalves
Supervisor: Prof. Carlos António Roque Martinho
Member of the Committee: Prof. Hugo Miguel Aleixo Albuquerque Nicolau

October 2018
Acknowledgments

First and foremost, I would like to thank my dissertation supervisor, Prof. Carlos Martinho, for all his work and patience along this journey, all the time invested in me, this work, and our brainstorming. I would also like to thank for all the motivation and strength to surpass several obstacles along this path, guidance and availability. On a more personal note, thank you for all the conversations we had about games and life in general.

The second person I would like to thank is my great friend Diogo Rato, thank you for all the support and motivation, for keeping me in the right path and pushing me to do better and keep moving forward.

To all my friends for their friendship, companionship and always being there when I needed the most. Thank you for all your support over all these years.

Last but not least, I would like to thank my family, thank you for all your love and caring throughout this journey, for believing in myself and my work, for giving me the support I needed to continue. A special thanks to the love of my life, my sister, for making me aim higher and giving me the motivation and strength I needed. To all of you, I am deeply thankful!
Abstract

There are hundreds of millions of people that play computer games daily and, although they have different needs, these people acknowledge that content (be it monsters, items or even doors) is one of the most important parts of a game. We firmly believe that placing content personalized to each person will improve the overall player experience, but how do we make content placement tailored to individual characteristics? This work proposes a computational model capable of placing content taking the player’s model into account by using an interest curve to place and pace the content. This model places, procedurally, the challenges according to the player’s model and progression of these challenges along the level. With this model, different players with different models would be playing the same game, while getting content directed to them. Our model was implemented having Legend of Grimrock 2 as a game environment, since it provides a mod-friendly editor and has the level’s information stored in an easy to access text file. Before testing our model’s content placement, a challenge validation was made, where we concluded that the designed challenges were being understood as intended. Final results indicated that the ordering of challenges according to the curve of interest appropriate to the player’s model only affects the positive emotions reported by the player, but does so in a very clear way. That is, choosing the moment when the challenges most valued by the player appear had no impact on the less positive aspects of the experience but amplified the positive moments.

Keywords

Video games, procedural content generation, procedural content placement, player model, player experience, interest curves
Resumo

Existem centenas de milhões de pessoas que jogam jogos de computador diariamente e, apesar de terem diferentes necessidades, estas pessoas reconhecem que o conteúdo (seja monstros, items ou portas) é uma das partes mais importantes de um jogo. Nós acreditamos firmemente que colocar conteúdo personalizado para cada pessoa irá melhorar, no geral, a sua experiência de jogador, mas como colocamos conteúdo adaptado às características individuais? Este trabalho propõe um modelo computacional capaz de colocar conteúdo tendo em consideração o modelo de jogador, usando uma curva de interesse para filtrar e colocar o conteúdo. Este modelo coloca, de forma procedimental, os desafios tendo em conta o modelo de jogador e a progressão destes desafios ao longo do nível. Com este modelo, diferentes jogadores com diferentes modelos jogariam o mesmo jogo, tendo conteúdo direcionado para eles. O nosso modelo foi implementado usando o Legend of Grimrock 2 como ambiente de jogo, uma vez que este nos fornece um editor fácil de usar e tem a informação dos níveis guardada num ficheiro de texto de fácil acesso. Antes de testarmos a colocação de conteúdo do nosso modelo, fizemos uma validação dos desafios, onde concluímos que estes estavam a ser entendidos como pretendido. Os resultados finais indicaram que a ordenação dos desafios de acordo com a curva de interesse adequada ao modelo do jogador apenas afeta as emoções positivas relatadas pelo jogador, mas fá-lo de uma forma bem clara. Ou seja, escolher o momento em que os desafios mais valorizados pelo jogador aparecem não teve impacto nos aspectos menos positivos da experiência mas amplificou os momentos positivos da mesma.

Palavras Chave

Video jogos, geração de conteúdo procedimental, colocação de conteúdo procedimental, modelo de jogador, experiência de jogador, curvas de interesse
Contents

1 Introduction .............................................. 1
   1.1 Motivation ........................................... 3
   1.2 Problem ............................................... 4
   1.3 Hypothesis ............................................ 4
   1.4 Objectives ............................................ 5
   1.5 Contributions .......................................... 5
   1.6 Document Outline ..................................... 6

2 Related Work ........................................... 7
   2.1 Personality Models .................................... 9
      2.1.1 Myers-Briggs Type Indicator ................. 9
      2.1.2 Five Factor Model .............................. 9
      2.1.3 Cloninger's Temperament and Character Inventory .... 10
   2.2 Player Models ....................................... 11
      2.2.1 Bartle Player Types ............................ 11
      2.2.2 Demographic Game Design .................. 12
      2.2.3 BrainHex ....................................... 13
      2.2.4 Quantic Foundry's Gamer Motivation Model .... 14
   2.3 Procedural Content Generation .................... 15
      2.3.1 Experience-Driven Procedural Content Generation ... 16
      2.3.2 3Buddy ....................................... 17
   2.4 Progression .......................................... 18
      2.4.1 Façade ....................................... 19
   2.5 Discussion ........................................... 19

3 Computational Model ................................. 21
   3.1 Legend of Grimrock 2 .................................. 23
      3.1.1 Content ....................................... 24
      3.1.2 Level Editor ................................... 24
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Bartle Player Types and Demographic Game Design side by side</td>
<td>12</td>
</tr>
<tr>
<td>2.2</td>
<td>EDPCG main components</td>
<td>16</td>
</tr>
<tr>
<td>2.3</td>
<td>3Buddy’s computational approach</td>
<td>18</td>
</tr>
<tr>
<td>2.4</td>
<td>Façade story tension value arc, used for scoring beats in beat sequencing</td>
<td>20</td>
</tr>
<tr>
<td>3.1</td>
<td>Legend of Grimrock 2</td>
<td>23</td>
</tr>
<tr>
<td>3.2</td>
<td>Legend of Grimrock 2 level editor</td>
<td>25</td>
</tr>
<tr>
<td>3.3</td>
<td>Legend of Grimrock 2 map components</td>
<td>26</td>
</tr>
<tr>
<td>3.4</td>
<td>Challenges’ content</td>
<td>28</td>
</tr>
<tr>
<td>3.5</td>
<td>Achiever</td>
<td>28</td>
</tr>
<tr>
<td>3.6</td>
<td>Conqueror</td>
<td>29</td>
</tr>
<tr>
<td>3.7</td>
<td>Daredevil</td>
<td>29</td>
</tr>
<tr>
<td>3.8</td>
<td>Mastermind</td>
<td>30</td>
</tr>
<tr>
<td>3.9</td>
<td>Seeker</td>
<td>30</td>
</tr>
<tr>
<td>3.10</td>
<td>Survivor</td>
<td>31</td>
</tr>
<tr>
<td>3.11</td>
<td>Interest curve</td>
<td>32</td>
</tr>
<tr>
<td>3.12</td>
<td>Original and modified side by side</td>
<td>33</td>
</tr>
<tr>
<td>3.13</td>
<td>Model’s architecture</td>
<td>34</td>
</tr>
<tr>
<td>3.14</td>
<td>Interface</td>
<td>36</td>
</tr>
<tr>
<td>3.15</td>
<td>Interface in testing mode</td>
<td>37</td>
</tr>
<tr>
<td>3.16</td>
<td>Replication of the BrainHex questionnaire</td>
<td>37</td>
</tr>
<tr>
<td>4.1</td>
<td>Visualization of the challenge’s pattern in Listing B.1</td>
<td>43</td>
</tr>
<tr>
<td>4.2</td>
<td>“Stamp” examples</td>
<td>43</td>
</tr>
<tr>
<td>4.3</td>
<td>Content being displayed</td>
<td>45</td>
</tr>
<tr>
<td>4.4</td>
<td>New level displayed on editor</td>
<td>46</td>
</tr>
<tr>
<td>5.1</td>
<td>Competence component</td>
<td>58</td>
</tr>
</tbody>
</table>
5.2 Sensory and imaginative immersion component ................................. 59
5.3 Flow component ............................................................................. 60
5.4 Tension component ........................................................................ 60
5.5 Challenge component ...................................................................... 61
5.6 Negative affect component ............................................................. 61
5.7 Positive affect component ............................................................... 62
List of Tables

5.1 Challenge validation’s processed data ........................................... 53
5.2 Second iteration with 4 participants ............................................. 55
5.3 Summary of the final evaluation results ....................................... 63

List of Algorithms

1 Content placement algorithm’s run ............................................... 33
2 Player interest calculation ............................................................ 42
3 Content sorting ........................................................................... 44
# Listings

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1 Example of a player's model</td>
<td>31</td>
</tr>
<tr>
<td>3.2 Example of an interest curve's file</td>
<td>38</td>
</tr>
<tr>
<td>4.1 Player interest dictionary entry example</td>
<td>42</td>
</tr>
<tr>
<td>A.1 Legend of Grimrock 2 dungeon code in Lua</td>
<td>73</td>
</tr>
<tr>
<td>B.1 Example of a challenge</td>
<td>81</td>
</tr>
<tr>
<td>C.1 Information for archetypes’ values calculations</td>
<td>83</td>
</tr>
</tbody>
</table>
# Introduction

## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 Motivation</td>
<td>3</td>
</tr>
<tr>
<td>1.2 Problem</td>
<td>4</td>
</tr>
<tr>
<td>1.3 Hypothesis</td>
<td>4</td>
</tr>
<tr>
<td>1.4 Objectives</td>
<td>5</td>
</tr>
<tr>
<td>1.5 Contributions</td>
<td>5</td>
</tr>
<tr>
<td>1.6 Document Outline</td>
<td>6</td>
</tr>
</tbody>
</table>
1.1 Motivation

According to M. Hendrikx et al. [1], hundreds of millions of people play computer games every day. M. Hendrikx et al. also states that for these people, game content plays a major entertainment role. So far, game content was created and placed manually by designers and that was sufficient to ensure that the quality and quantity of game content matched the demands of the playing community, but, over the past decade, that fact is changing. More and more video game companies are shifting towards procedural content generation [2].

Since the appearance of procedural content generation, there have been numerous studies and applications related to game content in actual video games, such as, generation of dungeon levels in Rogue, galaxies and planets in Elite, maps in Civilization IV, animations in Spore, world and content in Minecraft, weapons in Galaxy Arms Race [3], texture and terrain in Tiny Wings and type, number and placement of items and monsters in Diablo.

Despite all the diverse existing applications in actual games and the hundreds of millions of people playing every day, content is, normally, placed while following an already determined progression, without ever taking its relevance to the player into consideration.

To our knowledge, there is not an application where different content is selected based on a model of player types and then placed in a game, providing personalized content to a specific type of player. Games are normally created focusing in a specific type of player. By inserting content personalized to a player, games that focus certain player types would attract more player diversity, making it more enjoyable for everyone. That way, different players with different models would be playing the same game, while getting content directed to them.

Improving games and their experience can also be done by the use of story arcs. Some games use story arcs in order to provide a controlled story to a player, specifying when a climax will occur. Through the specification of story arcs, games can provide a desired tension to the player, triggering different emotions while playing. The specification of these arcs is also used to provide a progression to the player. Games’ stories control the tension’s progression along its course and could serve as inspiration to games in general.

Therefore, this is where our idea of placing content adapted to a player’s profile comes in. We will place the content by crossing it with an interest curve, making the placement more or less relevant to each player in a specific moment. By using an interest curve as a placement rule, we will be able to...

---

2. D. Braben and I. Bell, “Elite”, Acornsoft, Firebird, Braben, David and Bell, Ian, 1984
3. S. Meier and S. Johnson, “Civilization IV”, Firaxis Games, 2005
8. Modified story arc with distance as the XX axis and interest as the YY axis
control the content given to a player. We believe that our idea will be able to provide a content placement
to the player’s taste and an overall greater experience.

To achieve the aforementioned, we chose Legend of Grimrock 2 as the game to use with our model,
since it provides a mod-friendly level editor and its dungeons are written in a text file which allow us to
easily rewrite them as we please, making it the game for our idea.

1.2 Problem

Content is one of the most important components in a game and so is its placement. A good and
thought content placement can make a game better than it already is. Content placement is crucial for
level design, since it would be less engaging to play a level without any type of content, be it monsters,
items or even doors.

There are numerous ways to place content, and they range from a completely random to a well
thought placement. On one extreme, the completely random way [4], where there are no restrictions
and content can be placed anywhere. By placing content in a completely random way, it can appear
several times in the same location, not needing to make sense or follow any rule. On the opposite
extreme, the well thought placement, where designers follow a set of rules and spend time thinking on
what content to place and where to place it, aims to improve the game. Level designers, normally, try to
transmit a sensation of progression by their content placement, through content placing while following
a story, or just by constantly increasing the game difficulty.

Above, were referred numerous ways to place content, but they all fall in two categories: content
placed by hand and by procedural content generation. At first sight, it may seem that content placed by
hand provides a better player experience, since it is done by level designers after a meticulous thinking
process and it is done for a specific level to provide a, already defined, sensation of progression. How-
ever, it is challenging to adapt this content to different people, for that same reason. On the other hand,
content placed by procedural content generation has a higher adaptation probability, as one can insert
different parameters to control the placement and type of content, yet, procedural content generation
algorithms are somewhat “blind” when it comes to taking people personality into account.

So, how do we make content placement tailored to the player’s individual characteristics?

1.3 Hypothesis

Assuming we know the player model a priori and can provide distinct content to cover each player’s
preferences, the content placement will provide an overall greater experience to the player.

That being said, we propose to create an algorithm to make content placement, following a given
interest curve, so that each content element has its relevance for the player, taking the player's model into account. In order to create this algorithm, we will select an architecture that better suits our problem, so that we can use the player model results and interest curve as parameters, as we need them to make the content placement relevant to the player, and a player model, in order to obtain a profile to be taken into account.

In short, we consider the strengths of a content placement algorithm that takes the player's likes into consideration, making the content suited for his profile.

So, with this being said, our intention is to find if using an interest curve to place and pace content based on its relevance to the player's interest will help creating a better game experience for the player.

1.4 Objectives

Our main objective with this work is to test our hypothesis which is finding out if using an interest curve to place and pace content based on its relevance to the player's interest will help creating a better game experience for the player. In order to achieve this, we developed a computational model that, after receiving a level, the player's model, a list of challenges and an interest curve as parameters, chooses the challenges that best match the interest curve and places them throughout the level, making it a content placement taking the player's model into account.

1.5 Contributions

With the conclusion of our work, we ended up with the following contributions:

• State of the art in procedural content generation, personality and player models.

• Computational model for placing content throughout an empty level, taking a player's model into account.

• Computational model capable of reading challenges' patterns and searching them on the level.

• Grammar for challenge interpretation.

• An interface for a better interaction with the model.

• A study to test if our model's content placement was able to improve a player's game experience.
1.6 Document Outline

In this chapter, we presented our motivation for this work, explaining why procedural content generation, personalized content, players' models and story arcs are interesting approaches and how they could make games better. Then we talked about different content placement techniques, leading to the question, and our problem, how do we make content placement tailored to the player's individual characteristics? and stated that our intention is to find if using an interest curve to place and pace content based on its relevance to the player's interest will help creating a better game experience for the player. Afterwards, we presented our contributions with this work.

In the second chapter, Related Work, we describe three relevant personality models: Myers-Briggs Type Indicator, Five Factor Model, also known as Big Five Personality Traits, and Cloninger’s Temperament and Character Inventory. We also present four player models: Bartle Player Types, Demographic Game Design, BrainHex and Quantic Foundry’s Gamer Motivation Model. Afterwards, we go through procedural content generation in more detail, explaining what it is, its uses and the taxonomy used to evaluate and classify PCG. We, then, refer two works that use PCG: 3Buddy and Experience-Driven PCG. After PCG, we refer progression, presenting Façade, an artificial intelligence-based research experiment in electronic narrative. We end this chapter with a discussion, where we summarize all that was presented, concluding on what to apply in our work and why.

The third chapter, Computational Model, is where we describe the four components that our model uses: challenge library, player model, interest curve and level and present our model's architecture. We start by presenting Legend of Grimrock 2 in order to introduce the challenges and how we came up with their grammar. After this, we present the player model we used and how we were able to calculate each archetype’s value. Afterwards, we show the interest curve used to pace and place content and explain how it was created. We, then, present the chosen level and the modifications we made. In the next section, we present our model's architecture, describing its behavior and showing the interface we created to allow an easier interaction with the model.

Model Implementation, Chapter four, is where we present our algorithm, the responsible for content placement. We start by explaining how we represented the challenges, and go through our algorithm's run, step by step. We end this chapter by describing some design decisions.

In the fifth chapter, Evaluation, we describe both our tests, challenge validation and final evaluation, presenting each of these tests’ samples, measurements, procedures and results. We end this chapter with a discussion about the obtained results.

The sixth, and final, chapter, Conclusion, as the name suggests, is where we draw our final conclusions regarding the developed work and present some potential future work.
Related Work

Contents

2.1 Personality Models .................................................. 9
2.2 Player Models ......................................................... 11
2.3 Procedural Content Generation .................................. 15
2.4 Progression .......................................................... 18
2.5 Discussion ............................................................ 19
2.1 Personality Models

A person’s personality is directly related to the way he/she acts upon the world. Each person has different behavior, reasoning and characteristics, making them unique. Having stated the previous, we assume different players have different needs. Some might only play in a competitive environment, challenging others, proving they are better, at a logical or mechanical level, and trying to climb to the top, while others might play to end a game’s story or explore the world to collect all collectibles in a game. In this sub-section we will explain the psychological theories of Myer-Briggs Type Indicator (MBTI), Five Factor Model (FFM), also known as Big Five Personality Traits, and Temperament and Character Inventory (TCI), that we consider relevant to our work.

2.1.1 Myers-Briggs Type Indicator

Myers-Briggs Type Indicator or MBTI [5] is an introspective self-report questionnaire with the purpose of indicating differing psychological preferences in how people perceive the world around them and make decisions based on those perceptions. MBTI was constructed by Katherine Cook Briggs and her daughter Isabel Briggs Myers and is one of the most used psychological models in the interactive entertainment field [6]. This psychological theory is based on the conceptual theory proposed by Carl Jung [7], a Swiss psychiatrist and psychoanalyst, who had speculated that humans have four main psychological functions of consciousness by which they experience the world: sensation (perception by means of immediate apprehension of the visible relationship between subject and object), intuition (perception of processes in the background), thinking (function of intellectual cognition) and feeling (function of subjective estimation) and that one of these four functions is dominant for a person most of the time. These four main functions could then vary according to two attitude types: extraversion and introversion, making a total of eight dominant functions. MBTI went further and added two new psychological functions, judging and perceiving, to explain the interaction with the outside world, doubling the possible personality types.

Although one of the most used psychological models in the field of interactive entertainment, MBTI exhibits significant psychometric deficiencies, notably including poor validity and reliability, having already been questioned in the field of psychology [8].

2.1.2 Five Factor Model

Five Factor Model or FFM [9], also known as Big Five Personality Traits, defines personality through five different factors or personality traits based on a linguistic analysis. These five factors have been defined as openness to experience, conscientiousness, extraversion, agreeableness and neuroticism, often represented by the acronyms OCEAN or CANOE. Openness to experience is a trait related with the
appreciation of art, emotion, adventure, unusual ideas, imagination, curiosity and variety of experience. People with a high score in this have an intellectual curiosity always searching novelty. Conscientiousness implies a desire to do a task well and to take obligations to others seriously. Conscientious people are self-disciplined, careful and vigilant. They prefer to have a plan rather than act on an impulse. Extraversion trait is marked by pronounced engagement with the external world. People with a high score in this trait tend to look for others and being outgoing. They possess high group visibility, like to talk and assert themselves. Agreeableness is a personality trait manifested in a person’s ability of being kind, sympathetic, cooperative, warm and considerate. Agreeable people have an optimistic view of human nature. Neuroticism is related with the easy development of undesirable emotions such as anger, anxiety and fear. People with high neuroticism indexes are at risk for the development of common mental disorders like anxiety and mood disorders.

The Five Factor Model has uses in numerous areas for diverse reasons. There are uses in the study of personality disorders [10] (where there are several published studies relating these disorders to FFM), common mental disorders [11], health [12], education [13] (where studies are related to academic achievement), work [14] (relating success rate in work to FFM) and romantic relationships [15] (defining the various types of couples).

This model is well known and has a vast usability, however, there are a considerable amount of critiques to it about its limited scope, methodological issues and theoretical status. Dan P. McAdams [16] has referred to Big Five as a “psychology of the stranger”, since it only identifies traits that are easily observed in a stranger, lacking the ability to observe aspects of personality that are more privately held or more context-dependent. Was also referred by Hans J. Eysenck in [17] that a five factor solution depends on the analyst’s interpretation skills and that a larger number of factor may underlie these five factors. Lastly, Hans J. Eysenck also mentions the underlying causes behind the five factors being unknown.

2.1.3 Cloninger’s Temperament and Character Inventory

Cloninger’s Temperament and Character Inventory or TCI [18] appears as a 240-item questionnaire designed to explain the unique personality of an individual, identifying the intensity of and relationships between seven personality dimensions, four of which being Temperament and the other three being Character. Temperament refers to the automatic emotional responses to experience and character to the differences in goals and values, which influence choices and intentions. Temperament’s four personality dimensions are: Novelty Seeking, Harm Avoidance, Reward Dependence and Persistence. Novelty Seeking is observed as exploratory activity in response to novelty, impulsiveness, extravagance in approach to cues of reward, and active avoidance of frustration. People with high Novelty Seeking tend to be quick-tempered, curious, easily bored, impulsive, extravagant, and disorderly. Harm Avoidance is
seen as pessimistic worry in anticipation of problems, fear of uncertainty, shyness with strangers, and rapid fatigability. Individuals high in Harm Avoidance are fearful, socially inhibited, shy, passive, easily tired, and pessimistic even in situations that others do not worry about. Reward Dependence relates to sentimentality, social sensitivity, attachment, and dependence on others' approval. People with high values in Reward Dependence are tenderhearted, sensitive, socially dependent, and sociable. Lastly, Persistence dimension is related to industriousness, determination, and perfectionism. Highly Persistent people are hard-working, perseverant, and ambitious overachievers. The three Character’s personality dimensions are: Self-Directedness, Cooperativeness and Self-Transcendence. Self-Directedness quantifies the extend to which an individual is responsible, reliable, resourceful, goal-oriented, and self-confident, also known as “willpower”. Cooperativeness is a personality trait concerning the degree to which individuals conceive themselves as integral parts of human society. The last of the three Character’s personality dimensions, Self-Transcendence, is associated with experiencing spiritual ideas such as considering oneself an integral part of the universe.

TCI was afterwards revised [19] and gave place to TCI-R, which assesses the same Temperament and Character domains as the TCI, but included further development and refinement of the Persistence Temperament domain. TCI-R had another change, and that was the exchange from a “true/false” format to a 5-point Likert scale format. They also reformulated 51 of the 240 items from the previous TCI.

This model is still vastly used in neurobiological studies [20].

2.2 Player Models

Players tend to have preferred game genres. These choices are normally made based on the game's content. Players usually play games that have content tailored to their likes. Their likes derive from the type of players they are. Player models try to explain these preferences. In this sub-section we will present and explain four player models, that have as objective the categorization of players by type. Three of these player models were created based on some of the previously explained personality theories and are Demographic Game Design (DGD), BrainHex and Quantic Foundry’s Gamer Motivation Model. The fourth model, Bartle Player Types, was based on the direct observation of players.

2.2.1 Bartle Player Types

Numerous player models have been proposed and debated over the years. One of the earliest was the Bartle Player Types [21]. This model proved to be one of the most referenced and enduring. Richard Bartle, through MUDs (Multi-User Dungeon, text-based adventure game with no graphics at all), conducted a study used to classify players according to their preferred in-game actions. The study analysis showed four different pattern types that lead to four different characters: Achievers, Explorers, Socialis-
ers and Killers. These different characters can be represented in a two-dimensional graph, containing Players-World as the X axis and Acting-Interacting as the Y axis, as shown in Fig. 2.1.

Achievers, also known as Diamonds, are players that act upon the world and they like competition. These players enjoy beating difficult challenges set by the game environment or themselves. The harder the goal, the better. Achievers tend to feel the most rewarded after surpassing a harder challenge. Explorers, dubbed Spades for their tendency to dig around, are players who interact with the world. They enjoy exploring the world, not just the map but also the game mechanics. These players tend to know all the mechanics, short-cuts, tricks and glitches there are to know in a game. Socialisers, or Hearts, are known to interact with players. In a game, they are more interested in having a conversation or creating bonds with players than playing the game itself. Socializers are, normally, involved in the community, spreading knowledge to other players. Lastly, Killers, referred as Clubs, enjoy to act on players. They prosper in competition and prefer to play against real players than scripted enemies (NPCs, non-player characters), demonstrating superiority.

2.2.2 Demographic Game Design

Chris Bateman proposed a player model, Demographic Game Design or DGD [22], based on applying Myer-Briggs typology to data gathered about the players and their gameplay needs. This model focuses on market oriented game design. DGD has some similarity with Bartle Player Types, identifying four player types: Conquerors, Managers, Wanderers and Participants (Fig. 2.1). Conquerors (Type 1) are players concerned with competition and beating games. Managers (Type 2) are strategists and players interested in management gameplay. Wanderers (Type 3), as the name implies, are players who like to wander the game, who enjoy an open game. Participants (Type 4) are the type of players that focus on the play’s emotional context.
These four types are subdivided into Hardcore (H1, H2, H3 and H4) and Casual (C1, C2, C3 and C4), making eight total subtypes. These subdivision makes this model more focused on the players’ abilities instead of on their likes. Using this model, players are usually identified as a percentage of each type and not just belonging to one.

2.2.3 BrainHex

BrainHex is a player satisfaction model that was created by Nacke et al. [23] and is based on the results from the earlier DGD and neurobiological studies. Although BrainHex is based on neurobiological studies, it does not use any neurobiological evaluation technique, but questionnaires instead. This model divide players into seven different archetypes: Seeker, Survivor, Daredevil, Mastermind, Conqueror, Socialiser and Achiever. Seeker is the type of player to roam a game world just for the sake of curiosity and wonder. Survivor relates to the enjoyment of a frightening experience making a player resort to survival instincts. A player whose play style involves thrill, excitement and risk, constantly playing on the edge is identified as a Daredevil. Mastermind is an archetype that refers to puzzle solving and strategic thinking, surpassing harder challenges while making the most efficient decisions. Conquerors are players that enjoy a good struggle. These players like to challenge foes that others do not dare and show superiority, dueling and beating other players. Socialiser is related to social interaction and refers to players that enjoy being in community, be it by helping, talking to, or hanging around with others. Achiever is a goal-oriented archetype. Players identified as achievers enjoy checking check boxes, obtaining their satisfaction from attained goals or achievements. Players are usually identified by one main archetype and a secondary one. A player can still have exceptions attributed to his model results, depending on what were his answers while doing the test. Exceptions describe what a player dislikes about playing games. There are seven different exceptions - each exception is related to one of the archetypes - which are: No Commitment, No Mercy, No Punishment, No Problems, No Pressure, No Fear and No Wonder. No Commitment refers to a player that dislikes being asked to fully complete everything and is the exact opposite of the Achiever archetype. No Mercy, as the opposite of the Socialiser archetype, is related to players that do not care about playing with others. Players that are attributed the No Punishment exception dislike repeating the same task and struggling to overcome stronger foes or seemingly impossible challenges. No Problems is related to players that do not enjoy to solve puzzles or work out solutions on their own. The No Pressure exception refers to players that dislike to perform under pressure. No Fear, as the name implies, is related to players that do not enjoy the feeling of fear. Lastly, players that are attributed the No Wonder exception prefer clearly defined tasks instead of being asked to search for things.

Nacke et al. [23] state that DGD’s Conqueror, Manager, Wanderer and Participant types correspond broadly to BrainHex’s Conqueror, Mastermind, Seeker and Socialiser.
2.2.4 Quantic Foundry’s Gamer Motivation Model

Nick Yee and Nicolas Ducheneaut, Co-Founders of Quantic Foundry, came up with an empirical Gamer Motivation Model [24] through the gathering of data from over three hundred thousand gamers and using a psychology research standardized process that revolves around factor analysis (a statistical technique that identifies how variables cluster together), which is used by many other models, including the FFM. The Gamer Motivation Model, unlike other models that are based on the MBTI, is explained in light of what you would expect looking at FFM. This model identifies twelve motivation factors: Destruction, Excitement, Competition, Community, Challenge, Strategy, Completion, Power, Fantasy, Story, Design and Discovery. Destruction refers to the enjoyment of explosions and chaos. Excitement is the constant search for an adrenaline rush in fast-paced and intense environments. Competition, as the name implies, is the enjoyment of competition with other players. Community refers to the need of interaction and collaboration with other players. Challenge is the preference for games of skill and harder challenges. Strategy requires logical and strategic thinking, as it refers to the enjoyment of careful decision-making in games. Completion is the desire to fully complete a game (complete every mission or task, acquire every collectible and discover hidden content). Power refers to the need of fast acquiring power, making the player powerful within the game world context. Fantasy is the player’s ability to escape reality to be someone else, somewhere else. Story refers to the search of a complex storyline and intriguing characters. Design is the need for expression and customization. Lastly, Discovery is the desire to explore and experiment the game world.

These twelve factors can be clustered into six groups, as there are six pairs of closely related motivations. Destruction and Excitement are grouped into Action, since players with high Action scores like dramatic effects and visuals and be surrounded by enemies. Competition and Community are clustered into Social. Social players like to interact with other players, be it while collaborating or competing. The Mastery cluster is achieved by grouping Challenge and Strategy. Gamers with high scores on Mastery are masters on challenging strategic games. Grouping Completion and Power, they acquired Achievement, a cluster for players that like to collect power and collectibles. Fantasy and Story are grouped into Immersion, interesting storylines and customization options so gamers can deeply immerse in alternate worlds. Lastly but not least, Design and Discovery are clustered into Creativity. Gamers with high Creativity scores are constantly altering their game worlds, tailoring them to their own tastes.

The aforementioned six groups can also be clustered into three high-level groups: Immersion-Exploration, Achievement-Mastery and Action-Social. Yee writes that Immersion-Exploration “covers different ways of relating to the story and design of the game world”, Achievement-Mastery “covers different ways of progressing through and attaining power within the construct of the game world” and Action-Social “covers more energetic and gregarious modes of gameplay, seeking out arousing gaming experiences”.

14
2.3 Procedural Content Generation

Togelius et al. [25] describe Procedural Content Generation (PCG) as the algorithmic creation of game content with limited or indirect user input. In other words, the creation of game content through a software, with the help, or not, of a human being. PCG is being more and more used by companies to create their game content, as it solves one of the biggest problems in video game production, the cost of content creation. Through the creation of an algorithmic solution, companies decrease the need for manual labor, consequently decreasing the costs it entails. Although the cost reduction, this is not a solution chosen by companies on a daily basis, since it still lacks the ability to compete with level designers in terms of player experience creation.

According to Togelius et al. [25], PCG can be classified and evaluated through seven pairs of aspects: online versus offline, necessary versus optional, random seed versus set of parameters, generic versus adaptive, stochastic versus deterministic, constructive versus generate-and-test and automatic generation versus mixed authorship. Online refers to the generation of content while the game is being played, while offline, as the obvious opposite, is content generation while the game is not built, allowing changes from the designer. The necessary versus optional content are distinguished by the content importance. Necessary content is the content required to complete a level or unlock an important part of the game. Optional content is auxiliary content that can be discarded or exchanged without any negative effects. Random seed versus set of parameters are related to the control of content generation in PCG. Using a random seed the same content can be generated if the used seed is the same, while using a set of parameters we need, at least, an array of parameters that satisfy a set of specifications. Generic versus adaptive is related to the player behavior. Adaptive content generation, as the name implies, is the generation of content, taking the player behavior into account, as opposed to generic. Stochastic versus deterministic are distinguished by the ability of regenerating the same content. While deterministic PCG, given the same parameters, allows this regeneration, the same does not apply to stochastic PCG. Constructive versus generate-and-test differ in the iterations number. Constructive PCG is done in one pass, one iteration. Generate-and-test is iterated over and over again, alternating through generation and test phases, until a satisfactory solution is achieved. Lastly, automatic generation versus mixed authorship. These aspects are related to the input received. Automatic generation allows limited input in attempt to tweak parameters, while mixed authorship cooperates with a designer in the design process to generate a satisfactory solution.

There are numerous ways to use PCG. This being said, we will focus our work on a specific part of PCG, content placement. Next we will talk about three works somewhat related to ours: Experience-Driven PCG, 3Buddy and Façade. Experience-Driven PCG was chosen for its approach to create content and evaluate it using players’ feedback in order to optimize their experience. We selected 3Buddy for its application of PCG on an existing commercial game. Lastly, Façade belongs on our related work.
for its ingenious story beat scoring method.

### 2.3.1 Experience-Driven Procedural Content Generation

Yannakakis et al. define Experience-Driven PCG (EDPCG) [26], a framework for PCG, driven by user experience computational models. This framework is composed by four components: Player Experience Modeling (PEM), Content Quality, Content Representation and Content Generator, as shown in Fig. 2.2. Player experience is modeled based on game content and player. Since game content, when played, elicits player experience, it has to be tested in terms of quality and then, after searching the available, content that optimizes the player's experience is generated.

![Figure 2.2: EDPCG main components](image)

Yannakakis et al. identify three main approaches in Player Experience Modeling: subjective PEM (relies on data expressed by players), objective PEM (relies on data gathered from other types of player response) and gameplay-based PEM (relies on data obtained through the player's interaction with the game). Starting with subjective PEM, it can be based on free-response while playing, providing richer information about the player and his affective state, although harder to analyze, or on forced data gathered through questionnaires. This approach usually contains significant experimental noise (free-response) and can be intrusive with the insertion of questionnaires during gameplay sessions, being, however, sensitive to players' memory limitations. Objective PEM can be based on emotional models derived from emotional theories or model-free, where an unknown model between player input modalities and an emotional state representation via user annotated data is created. Lastly, gameplay-based PEM assumes that player actions and real-time choices are connected to player experience. As objective PEM, this approach can also be classified as model-based, model-free or a hybrid in between. Yannakakis et al. state that gameplay-based PEM is the most computationally efficient and least intrusive PEM approach of all three.

Content quality evaluation is done, using the acquired player models, in the content generation phase. The function of a content quality evaluator is to test a game content item, assigning a scalar to it, or vector
of real numbers, that indicates the suitability rate for use in the desired game and its capacity for inducing
the intended affective state. The Content Quality component is divided in three key evaluation functions:
direct, simulation-based and interactive. Direct evaluation function, as the name implies, is the extraction
of generated features, mapping them directly to a content quality value. Evaluation functions based on
simulation require the use of an artificial agent that plays through a game fragment that contains the
content being evaluated. These functions are distinguished by static and dynamic, where the distinction
resides in the assumption of an unchanging agent while playing the game (static) and the opposite
(dynamic), where content quality value somehow incorporates the changes. The last key evaluation
function is the interactive, which assigns value to content based on the player's interaction with it in the
game. Content is tested and fitness is evaluated during gameplay. Player data used to evaluate the
content can be collected in two different ways: explicitly and implicitly. Explicit data collection requires
the use of questionnaires or verbal input data. On the other hand, implicitly collecting player data is done
by measuring, e.g., the time spent interacting with game content.

Game content representation is a central question in EDPCG and can be symbolic or subsymbolic.
On one hand, Symbolically representing game content, within a tree or a graph data structure, allows
content generation in a designer controlled-fashion. On the other hand, subsymbolic representations,
e.g., artificial genotypes, allow a more diverse and innovative content creation.

After player experience being modeled, evaluation functions designed and content represented, the
content generator needs to search the available content by one that optimizes a particular player’s ex-
perience, ideally, identifying if content should be generated for that player.

2.3.2 3Buddy

Lucas and Martinho created 3Buddy [27], a GUI tool that works alongside the Legend of Grimrock 2's
level editor, giving the designer level altering suggestions. This tool produces suggestions based on
three different domains: innovation, guidelines and convergence. Innovation, as the name implies, is
the creation of something new, so this domain is in charge of suggesting content different from the one
made by the designer. The guidelines domain follows specific guidelines in order to generate content.
Lastly, convergence states how close the generated suggestion will be from the solution being created by
the designer. 3Buddy's interface consists, essentially, in three control sliders and an interactive canvas.
Each control slider is associated to one of the domains, allowing the designer to select the percentage
of a specific domain to be used in the suggestion creation. The canvas supports the interaction be-
tween 3Buddy and the level designer, allowing the later to select sections to import into the level editor.
3Buddy also provides different colors in the parts of the created suggestions that differ from the level
layout present in the Legend of Grimrock II editor. On 3Buddy's interface there is also a drop down menu
from where the designer can choose the goal he wants the guidelines domain to follow. A suggestion is
computed and presented in the interactive canvas when 3Buddy's behavior is triggered. Its behavior is triggered each time a dungeon layout is saved in the Legend of Grimrock II level editor or after an inactivity time-out. To compute and present a suggestion, 3Buddy uses three different pools of suggestions (illustrated by Fig. 2.3): convergence (set of suggestions evolved to be close to the current dungeon layout), innovation (set of suggestions evolved to be completely different from the current dungeon layout) and guidelines (set of suggestions evolved to follow the currently active goals).

![3Buddy's computational approach](image)

**Figure 2.3:** 3Buddy’s computational approach

So, explaining Fig. 2.3 in short, on one hand, innovation and guidelines pool are initialized with random dungeon individuals and are evolved to match their purpose. On the other hand, the convergence pool is initialized with copies of the current dungeon layout. Suggestion pool (new generation) is then created by selecting individuals from each pool, defining the percentage in the sliders. This new population is then transferred into the guidelines pool. After this process, a new cycle begins with only a difference, the guidelines pool is a mix of the previous three pools. This pool is evolved to find a new dungeon layout to suggest and, once selected, will be presented on the interactive canvas.

### 2.4 Progression

Pereira and Martinho created a progression model [28] that has the goal to evaluate the player’s skill level. This model uses a game’s mechanics (skills that can be executed by the player in order to over-
come a challenge), challenges (obstacles that are provided by the game) and pace (rate at which challenges are provided to the player) to test the player. Pereira and Martinho then use different levels of mastery to evaluate the player’s skill level on a specific mechanic, challenge or pace. A player initially has only one active mechanic and, after overcoming several challenges with that mechanic he masters it, unlocking a new mechanic that can be used with the first one, and so on. This way they can provide a progression to the player. In our work, we want the player to have a different kind of progression. We want the progression to be focused on the content placement taking the player into account. We want the player to have crucial moments in the game where he gets content completely compatible with his player model and other moments where content is the exact opposite of his model. We want the placement of content to follow an interest curve and in order to do that we have to score content and check its compatibility with the player’s model. This is why, next, we present Façade. Façade has a story beats scoring method that we find interesting and think it could help us in our work.

### 2.4.1 Façade

Façade [29], as defined by Mateas et al., is an artificial intelligence-based research experiment in electronic narrative. It is an attempt to create a real-time 3D animated experience where the player plays as Grace and Trip’s (an attractive and materially successful couple in their early thirties) long-time friend and becomes entangled in a high-conflict dissolution of their marriage, taking sides and being forced to make irreversible decisions while trying to solve the drama. Almost no direction is given to the player and his actions influence the occurring events, making them a big part on how the drama unrolls. Although Façade has a finite scope of possibilities\(^3\), the characters are designed to respond robustly to a variety of open-ended dialog from the player (questions and provocations included). Aside from the characters (Grace and Trip) and the player (human), Façade has a drama manager, an invisible agent, that continuously monitors the simulation and updates its rules, giving a well-formed overall experience to the player. These updates are organized into story beats\(^4\). The author annotates beats with preconditions and effects for specific story states, telling the drama manager when to use them. Afterwards, beats are scored based on an Aristotelian story tension value arc (illustrated by Fig. 2.4), and then selected based on which has the highest priority to the required part of the sequence.

### 2.5 Discussion

In the course of our research, we tried to reach an understanding on how players’ game content preferences affect their gameplay experience and how these preferences are connected to different types

---

\(^3\)Mateas et al. [29] state that players should feel they have exhausted the existing possibilities after playing 6 or 7 times

\(^4\)Beat is the smallest unit of dramatic action that moves a story forward

\(^5\)Façade [29] Figure 3
of players. To achieve this, we searched and explored numerous personality models and models of player types that we found relevant. We also researched works on procedural content generation and progression, discovering interesting applications that we could use as inspiration or guideline for our own work.

After completing the research and analyzing all the relevant findings, we selected the works we thought more appropriate to use and next we are going to explain why these were chosen. As model of player types, we opted for the utilization of Nacke’s et al. BrainHex [23], since it is a player satisfaction model and its results are absolute, meaning that a player’s model will be generated based only on the answers given to the BrainHex’s questionnaire and nothing else. By searching works on procedural content generation and progression, we found three works we acknowledge as relevant for our solution proposal. 3Buddy [27] will serve as inspiration for the design and creation of our GUI tool, as well as a guideline for our work, since it works in the same game environment we are going to use, Legend of Grimrock 2. EDPCG [26] provides an interesting content quality evaluation method using players’ interactions with game content in order to test its value and fitness, collecting player data through questionnaires. With this data we can access which content was preferred by the players. In Façade’s [29] drama management, story beats are used to store updates and annotations with preconditions and effects for specific story states. These beats are scored, using an Aristotelian story tension value arc, in order to give the highest priority to the ones that will provide the required tension for a specific part of the story sequence. Inspired by Façade’s [29] beat scoring method, we will use an interest curve\(^6\) to place content relevant to the player.

\(^6\)instead of tension, we cross the story arc with player’s interest or personality compatibility
3 Computational Model

Contents

3.1 Legend of Grimrock 2 ........................................................... 23
3.2 The Four Components ......................................................... 25
3.3 Model’s Architecture ......................................................... 33
In this chapter we describe with more detail our computational model and its components. We begin by talking about Legend of Grimrock 2, its content and editor. Then we explain all components needed for our model to work, starting with the challenges and how they were written, then we present the player model and how we incorporated it in our model and calculated the archetypes’ values, next we show the used interest curve and, lastly, the chosen Legend of Grimrock 2 level. Afterwards, we present our model’s architecture and explain how it all links together, describing the behavior and displaying the interface that allows the interaction with the model.

3.1 Legend of Grimrock 2

Legend of Grimrock 2\(^1\), shown in Fig. 3.1, is an action role-playing tile-based real-time dungeon crawler, developed by Almost Human Ltd. (a Finnish company established in 2011), having Dungeon Master\(^2\) as one of its inspirations. To advance in this game, the player has to solve different puzzle combinations, giving him the upper hand in terms of items and equipment, and engage in combat. He controls a party of one to four characters, from a first-person point of view, through a grid-based world. The party characters can be from one of the five following races: human, minotaur, lizardman, insectoid and ratling which affects their starting skills. Additionally each character has a class that grants them unique traits. The available classes in Legend of Grimrock 2 are: alchemist (brew potions and wield firearms), barbarian (raw power and speed), battle mage (can fight in the front row as well as cast spells

---

\(^1\)Almost Human, “Legend of Grimrock 2”, 2014
\(^2\)FTL Games, Victor Interactive, “Dungeon Master”, 1987
from the back row), farmer (familiar with digging ditches and plants’ growth cycles), fighter (master of close combat), knight (specialized in using armor and shield), rogue (uses ranged or light melee weapons) and wizard (uses enchanted staves and orbs to command great mystical powers).

3.1.1 Content

Legend of Grimrock 2 gives us diverse content which allows us to be creative, making it possible to do almost any type of challenge\(^3\). The game offers a variety of monsters, from ones with a basic attack move pattern to more challenging ones with different types of attack, ranges and speed. It also contains a panoply of weapons and armors. Each armor and weapon has its effects and, in addition, some weapons have basic and special attacks which provides a less monotonous gameplay to the player. Furthermore, the game has obstacles, wall traps, spike traps, pressure plates, doors, hidden doors, trap doors, fake walls, invisible walls, wall triggers, floor triggers, food, wall buttons, hidden wall buttons, lights, spells, books, teleporters, wall text, levers, locks, chests, altars, alcoves, etc. In addition to this list of content, each piece of content has its own attributes which gives us the freedom to personalize it at will and some of them still have connectors that let us link all together, making it easier to create challenges instead of just spawning random content in a room. The game also has script entities and timers which allow us to change some aspects and mechanics related to the characters and game itself.

3.1.2 Level Editor

In addition to the aforementioned, Legend of Grimrock 2 provides its players a mod friendly level editor, as shown in Fig. 3.2.

This level editor gives its users three main tools: select entities, asset browser and brush. Select entities (represented by the color red in Fig. 3.2), as the name suggests, is a selection tool that allows the user to point and click the level assets, selecting the chosen one. The asset browser tool (represented by the color green in Fig. 3.2) is, in its essence, a content menu, where the user filters and chooses which assets to insert into the level. From this menu, the users can select from a wide variety of content. The last tool, brush (represented by the color blue in Fig. 3.2), is the responsible for level creation. Brush gives the users a tile selection menu, from where they can freely choose the type of environment to create, be it a dungeon, a forest or even a beach. Besides the main level editing tools, the editor also provides testing tools (represented by the color purple in Fig. 3.2) that allow the designer to preview, play and debug the level being created.

\(^3\)Created by assembling several pieces of content in one room
3.1.3 Reasons for Chosen Environment

With all the aforementioned, Legend of Grimrock 2 was our chosen work environment, because it has a vast diversity in content and a mod friendly level editor, which makes it a lot easier to experience and assemble all content, giving us the power to create as many and diverse challenges as we want. In addition to that, it also gives us the dungeon layout and content in an easy to access text file, making our only work to rewrite this file following the original one, generating new and distinct dungeons. In Appendix A, we present an example of a dungeon layout - the one we used - written in an easy to access text file. To complement all these reasons, Legend of Grimrock 2 has a forum with script tutorials and examples\(^4\), as well as, scripting and function documentation and a git repository\(^5\) with active mod creators, working towards the creation of new and updated functions.

3.2 The Four Components

This first section is about all the components needed for our model to work. Without them there would be no content placement taking a player's model into account nor a game environment where we could test our model's results. The components described and explained in this section are: challenges or challenge library, player's model, interest curve and level.

---

\(^4\)http://www.grimrock.net/modding/
\(^5\)https://github.com/JKos/log2doc/wiki
3.2.1 Challenges and their Grammar

Starting with the challenges, our idea was to have a json file that would act as a library of challenges, so we had to find a way of writing our challenges that could later be interpreted and used by our model.

We started by looking at all the content available in Legend of Grimrock 2 as an individual challenge, e.g. a door was challenge, so was a broken jar and a spider. So, all the existing content in the game world was an individual challenge. We soon realized that this definition of challenge was dull and was no challenge at all. We brainstormed and arrived to our current definition of challenge, an assembly of individual pieces in a room - e.g. a room with monsters, teleporters, trap doors, etc - which made a lot of sense and, with this definition, we were able to create more interesting and exciting challenges to add to our library.

Having our definition of challenges, we had the idea of using patterns to make the most of the existing rooms in our level. To implement this idea we had to create a grammar that would represent each component of the game: walls, floor and content. We also needed grammar rules to separate rows and columns, since, in Legend of Grimrock 2, a level is defined by a 32 by 32 matrix and our rooms were matrices as well, as shown in Fig. 3.3.

![Legend of Grimrock 2 map components](image)

*(a) Level  (b) Room*

**Figure 3.3:** Legend of Grimrock 2 map components

So, taking all this into account and after a few iterations, we had reached our grammar’s final version. Next, we present and explain some examples of our final grammar.

X: Wall
O: Floor
?: Wall or Floor
#: Center of the pattern
/: Separates rows
;: Separates columns

Door[<name> <orientation> <attribute_0> <attribute_n>]
Chest[<name> <orientation> <attribute_0> <attribute_n> [<Item_0> <Item_n>]]
Timer[<name> <orientation> <attribute_0> <attribute_n> [<Connection>]]

where

<Item> : Item[<name> <orientation> <attribute_0> <attribute_n>]
[<Connection>] : [<X> <Y> <action> <reaction>]
[<Connection>] : [<X> <Y> <reaction>]

“W”, “O” and “?” mark walls, floor and either one, respectively. The “#” symbol was added, because we wanted to change the fact that in Legend of Grimrock 2 the X and Y coordinates start at the top left corner of the map, which made our patterns’ reading start always at the top left corner as well. So, this symbol acts like a center and makes the reading start on its position. The separation symbols - “/” and “;” - were added to represent a matrix in a way, since they act as rows’ and columns’ delimiters. Door, Chest and Timer are examples of the content that exists in Legend of Grimrock 2, <name> allows us to choose specific content from each type, <orientation> can have one of four values: top, right, bottom and left, and specifies the content’s orientation. For each type of content there are different attributes, therefore <attribute_0> ... <attribute_n>. Legend of Grimrock 2 has content that can contain other items - chests, alcoves, altars, etc - and connect to other elements - timers, pressure plates, locks, levers, etc. With this in mind, we added [<Item_0> ... <Item_n>] to represent the n items that could be in a container, where each item has the same syntax as the rest of the content, and [<Connection>]. Each connection has <X> and <Y> that represents where in the challenge’s pattern the content to connect is. There are two types of connection: one with an action-reaction pair (<action>, <reaction>) and another with just reaction (<reaction>). Action represents the available actions in the content that connects, e.g. pressure plate has 3 available actions: activate, deactivate and toggle. Reaction represents the available reactions in the content that is connected, e.g. a door has 3 available reactions: open, close and toggle. There are connections with just reaction, because there is content that has only 1 available action, so, in this case, there is no need to choose, e.g. monsters have only 1 action and that is dying.

With our final grammar set, we created six challenges which were later validated by users (we will talk about this in more detail in Chapter 5). This validation was used to check if the users could identify each
challenge as an archetype of the BrainHex player model (achiever, conqueror, daredevil, mastermind, seeker, and survivor). We ended up not doing a challenge to cover the seventh archetype, socialiser, because it requires the interaction with other players which is impossible in a single player game.

As a result of a well succeeded validation, we ended up with the challenges shown in the following figures. To understand each challenge, we present in Fig. 3.4 every used element.

![Figure 3.4: Challenges’ content](image)

The achiever challenge, shown in Fig. 3.5, contains a chest with items and a scroll with information about the challenge, pressure plates that when pressed by an item spawn an armor piece and a script that will play a music and display a text when the armor set is complete.

![Figure 3.5: Achiever](image)
In the conqueror challenge (Fig. 3.6) there is a magma golem with interesting and diverse attacks mechanics and three scripts that deal with general game mechanics like game over and food.

![Image](image1.png)

\[(a) \text{ Design} \quad \text{(b) Result}\]

**Figure 3.6**: Conqueror

The daredevil challenge, illustrated by Fig. 3.7, is based on speed and timing, containing several trap doors with control timers and a script that deals with the fall consequences.

![Image](image2.png)

\[(a) \text{ Design} \quad \text{(b) Result}\]

**Figure 3.7**: Daredevil

Next, on Fig. 3.8, we have the mastermind challenge with a combination puzzle between a lock, levers, pressure plates and secret walls.

In the seeker challenge, presented in Fig. 3.9, we have a glowing power gem which can be seen through a door, however the only way to get to the gem is by searching and pressing a secret button on
the wall and going through a hidden door.

The last challenge, survivor (Fig. 3.10), contains mimics, teleporters, pressure plates and spike traps.

These challenges (Figs. 3.5 to 3.10) were designed in a way that allows all of them to be placed at every point on the curve of interest. In Appendix B we have an example of a challenge written in our json file.
3.2.2 Player Model and Archetypes Calculations

As stated before, we used the BrainHex player model. As aforementioned, BrainHex categorizes players in seven different archetypes - achiever, conqueror, daredevil, mastermind, seeker, socialiser and survivor. Despite not having created a challenge to cover socialiser, we still took it into account while calculating each player’s model.

Although BrainHex provides an online questionnaire from which players can obtain their model, we still preferred to replicate it on our application, so there were no errors while copying the archetypes’ values from one side to the other, since we need those values for our model to work. Appendix C has all the information needed to replicate the BrainHex questionnaire.

When we save a player’s model, we are saving each archetype and its respective value. Next, we show an example of a saved model.

Listing 3.1: Example of a player’s model

```
Achiever: 9;
Conqueror: 12;
Daredevil: 9;
Mastermind: 20;
Seeker: 6;
Socialiser: 18;
Survivor: 1;
```
3.2.3 Interest Curve

As previously mentioned, we adapted the idea from Façade [29] and used a modified story arc, where our XX axis represents the game distance or Manhattan distance to the starting location and our YY axis the interest percentage that will be used by our model when searching in which challenges does the player have the nearest matching interest. Although our work supports different curves, in Fig. 3.11 we have the interest curve used during the final evaluation of our thesis.

![Interest curve](image)

**Figure 3.11**: Interest curve

This interest curve has 6 points that represent the 6 rooms in which the challenges are placed. The distance in each point is the Manhattan distance between the starting location and the top left corner of each room. This curve was made by using the last points in Façade’s story arc, Fig. 2.4, as inspiration. We start our curve with a mini-climax, the we drop and build up until we reach the final climax and drop again. This curve acts as our content selector and cites the player’s progression.

3.2.4 Chosen Level

To test our model, we had to use a predefined level layout. We could choose either creating a new one or adapt an existing one. We opted by the existing level layout adaptation. We started with an existing level, since it was already tested by developers, and adapted it to our needs. Fig. 3.12 shows both levels’ layouts side by side.

We first deleted all existing content, leaving only the layout. Then, we chose the location for the 6 needed rooms and we tried to make them with a similar design so it wouldn’t create any type of bias while players completed the challenges. Afterwards, we made a single path between the starting and ending locations and left some of the original dead ends, so the player had the opportunity to explore. Finally, we illuminated the main path, so the player could always find the correct one.
3.3 Model’s Architecture

In this section, we will address our model’s architecture, describing the behavior between our content placement algorithm and the four aforementioned components. Then, we present the created interface that allows an easier and more visual interaction with our model.

3.3.1 Behavior

Before explaining the behavior of our model and how each component is used by the content placement algorithm, we present our model’s architecture, illustrated by Fig. 3.13. This architecture shows the connections between our algorithm and the four components. To better understand these connections, we present Algorithm 1 that shows a run of our content placement algorithm.

Algorithm 1 Content placement algorithm’s run

1: procedure CONTENTPlacementAlgorithm(level, playerModel, challenges, interestCurve)
2:   distanceMatrix := dMatrix ← CalcCellsDistance(level)
3:   playerInterest ← PLAYERInterestCalc(playerModel, challenges)
4:   challengesPerCell ← PatternFinder(level, challenges)
5:   challengesPerDistance ← SORTContent(interestCurve, playerInterest)
6:   chosenPlacement ← CHOOSEContent(dMatrix, challengesPerCell, challengesPerDistance)
7:   SPAWNContent(level, chosenPlacement)
8: end procedure

As shown in Algorithm 1, our content placement algorithm uses the four aforementioned components.
throughout a run. Before our algorithm’s run starts, we import a level that is analyzed and parsed. This level is written in a text file - an example of a level’s file can be seen in Appendix A - which contains all the level’s relevant information: type of cells used, dungeon layout, width, height and content.

After having read the file and parsed its content, our algorithm searches for the cells where content can be placed - floor tiles - and the starting location. With the search completed, it calculates the distance between the starting location and each walkable cell (floor), storing the shortest distances for each one.

With the distances calculated, our algorithm moves to the next step and calculates the player’s interest in each challenge. To accomplish this, it uses the player’s model, acquired after players answer the replicated BrainHex questionnaire, and the library of challenges. In this step, it starts by reading the archetypes’ values from the player’s model and adapts these values by adding 10 to each one, changing the range from [-10, 20] to [0, 30]. This way we don’t have any more problems with the calculations due to negative numbers. After this, it calculates the player’s interest in each challenge by applying the two following mathematical formulas:

\[ W_0 = \frac{P_0}{\sum_i P_i} \]  \hspace{1cm} (3.1)

where \( W_0 \) is the weight of a player model archetype, \( P_0 \) is the value of an archetype and \( \sum_i P_i \) is the sum of all archetypes’ values.

\[ U_k = \sum_i W_i O_i \]  \hspace{1cm} (3.2)

where \( U_k \) is the player interest in that challenge and \( \sum_i W_i O_i \) is the sum of multiplication between the weight of a player model archetype and the respective challenge’s binary value.

34
Before explaining the next step, we have to clarify the definition of full-match. A full-match happens when the pattern of a challenge matches completely a fragment of the level's layout. In a simpler way, the pattern is used as a “stamp”, where the middle is applied to all walkable cells, and applied to a fragment of the level's layout with the same number of cells, e.g. if the pattern is a 5 by 5 matrix, the fragment of the level's layout has to be a 5 by 5 matrix as well. If the comparison between all the cells from the “stamp” and all the cells from the fragment returns true, then we have a full-match. With this definition out of the way, in the next step, our algorithm reads the pattern of every challenge and saves the ones that had a full-match alongside the cell where the pattern’s reading started, discarding the rest.

Proceeding to the next step, the content placement algorithm sorts the challenges, using the curve of interest as the one in charge of determining the player’s progression. It starts by verifying which challenges have the closest player's interest to match the interests’ values stated by the interest curve’s dots. To compare both values, the algorithm applies the following mathematical formula for each dot in the interest’s curve:

$$V_k = (U_k - d_k)^2$$  \( (3.3) \)

where \( V_k \) is the square difference of interests, \( U_k \) is the player’s interest in each challenge and \( d_k \) is the interest in the interest curve’s dot. Then, it sorts the challenges by \( \min_k V_k \) for each dot, making the first challenge, the optimal choice (\( V_{\text{min}} \)).

Afterwards, our content placement algorithm chooses the content which will be placed in the level. In this step, it starts by discarding all cells that don’t have a distance that match the distance values stated by the interest curve’s dots. Having filtered the cells per distance, the algorithm verifies if these cells contain any available challenges from where to choose. If not, these cells are discarded, otherwise, it chooses a cell, prioritizing the one with the least available challenges, in order to avoid leaving rooms without content in the level, e.g. if there are 2 cells with different distances, one with 1 available challenge and the other with more than 1. However, the later cell has that same challenge in the first position of the list from where to choose. If we don’t prioritize the cell with least available challenges, this will leave the room at that cell’s distance empty. With the cell selection completed, our algorithm just has to choose the challenge at the first position from all that cell’s available challenges.

Having all choices done, our algorithm’s last step is to place each challenge in the respective cells. To accomplish this, it goes through each element of the challenges and places them in the correct cells in order to create the room with the challenge. Then, our content placement algorithm updates these cells with the new content in the level.

After a complete run, there is only one task left to do, exporting this new level. The level’s exportation rewrites the level’s file, which is, then, read by the editor, allowing players to test our algorithm’s content placement results.
3.3.2 Interface

In the previous subsection, we presented our model’s architecture and the behavior between our content placement algorithm and all components. In this one, we will present the interface that allows an easier interaction with our model and gives a “face” to the behavior happening in background. Fig. 3.14 shows our interface.

As stated before, this interface allows an easier interaction with our model. We used this interface throughout the tests with players in order to have some visual feedback of what was happening and to make the players wait the least time possible between answering the BrainHex questionnaire and playing the level.

To test the users, we started by clicking the Mode tab in the menu strip, represented by the letter A in Fig. 3.14, and changing it to testing, hiding the relevant elements (Fig. 3.15) for our test from the users, in order to not create any kind of bias.

After, we asked the user to answer the questionnaire. To open the replicated BrainHex questionnaire, illustrated by Fig. 3.16, the user has to click BrainHex in the menu strip (A in Fig. 3.14) and Take the quiz, which generates an ID to preserve the user’s anonymity (H in Fig. 3.14).

Answering the questionnaire, modifies the values shown in B. These modifications are explained in Appendix C. Having the player’s model in our interface, we saved it by going to the BrainHex tab in A once again and clicking save. This action saves the player’s model in a text file, as shown in Listing 3.1.
Figure 3.15: Interface in testing mode

Then, and after changing the application to developing mode once again, we imported the level we were going to use for the test (Appendix A), which was displayed in the interface (G in Fig. 3.14). This panel allows us to check the elements being placed in each cell by hovering the mouse over it. It also allows to confirm the X and Y positions, and the distance of each cell. Importing the level adds a message to the logger (E in Fig. 3.14). This logger displays success and error messages to every major action done - importing a level, loading, saving and cleaning a interest curve, loading
and saving a player’s model. Importing the level also automatically imports the challenges displayed in the interface’s grid (C in Fig. 3.14). This grid shows each challenge of our library and the respective BrainHex archetypes. Clicking in each challenge will display its design in the panel represented by the letter F in Fig. 3.14. With the level and challenges imported and questionnaire answered, we just needed to import one more component, the interest curve (D in Fig. 3.14). To import the interest curve, we went to the Interest Curve tab in A and clicked load. Selecting and loading an interest curve displays it in the panel represented by the letter D. In this panel, we can modify the curve by adding more points, which can be done by checking the “Allow new points” check box, and changing each points’ Y value, which can be done by checking the “Allow Y modifications” check box. We can also save and clean the curve by going to Interest Curve’s tab. Saving an interest curve creates a text file with the following content:

**Listing 3.2: Example of an interest curve’s file**

```
42,73;
188,199;
213,126;
339,52;
450,9;
548,167;
-
9,65;
41,5;
46,40;
73,75;
97,95;
118,20;
```

where the first block (before “-”) represents the panel’s values and the second block (after “-”) represents those values converted into game distance and interest.

Having all components, we went to the Algorithm tab in A and clicked Run. This action triggers the behavior presented in Algorithm 1. Then, we exported the results by clicking the File button in A followed by Export, making this new level appear in Legend of Grimrock 2’s editor, allowing the players to test our algorithm’s content placement.
Model Implementation

Contents

4.1 Challenges' Representation ........................................... 41
4.2 Game Distance .......................................................... 41
4.3 Player’s Interest .......................................................... 41
4.4 Pattern Finder ............................................................. 42
4.5 Content Sorting ........................................................... 44
4.6 Content Selection ......................................................... 44
4.7 Content Spawning and Connecting ................................. 45
4.8 Dungeon File Rewriting ............................................... 46
4.9 Design Decisions ......................................................... 46
In this chapter, we explain how our solution was implemented and present our implementation step by step, describing all our algorithm’s process throughout a run. We start with how we chose to represent the Legend of Grimrock 2’s challenges for this project. We, then, talk about the game distance and what algorithm we used to calculate it. Afterwards, we explain how we calculated and saved the player’s interest in each challenge. Next, we go into more detail on how we analyzed the patterns, found matches and stored the cells and challenges. After, we explain how we sort the content and select it. Afterwards, we talk about how we spawn content and connect all elements. Finally, we explain how all content is rewritten in the dungeon file after being exported.

### 4.1 Challenges’ Representation

Our challenges representation was also done with the use of classes. We created a class that represents a single challenge object and has all the attributes present in the json file. Those attributes are name, pattern, seeker, survivor, daredevil, mastermind, conqueror, socialiser and achiever. We also created a function that gives some information about the type of challenge, e.g. the function returns Achiever if the achiever attribute is equal 1. It also gives information if it isn’t a pure challenge, e.g. returns AchieverDaredevil if both attributes are equal 1. We also created a class that represents a challenge library. This class only has a list where it stores all challenges read from the json file.

### 4.2 Game Distance

After having all components ready and clicked the run button, the first task our algorithm does is calculating the game distance for each cell. To calculate our game distance, we used an A* search algorithm since it is an algorithm that aims to find a path between a specific starting node and a given goal node, having the smallest cost. In our case, we substituted the nodes and cost by cells and distance traveled. The heuristic used with the A* search algorithm was the Manhattan distance, since our game only has horizontal and vertical directions. To calculate the game distance for each cell, our A* implementation had to be called once for every walkable cell in the map, having the starting position as starting cell and each cell as the goal cell. After the run, each cell stores the final and shorter distance. This action is only needed once, since the level’s layout remains the same throughout the process.

### 4.3 Player’s Interest

The second task done by our algorithm is calculating the player interest for each challenge. To accomplish this, our algorithm follows the process described by Algorithm 2.
Algorithm 2 Player interest calculation

1: procedure PLAYERINTEREST(archetypesValues, challenges)
2:   archetypesValues ← ADD(archetypesValues, 10)
3:   sum ← SUM(archetypesValues)
4:   archetypesWeights ← DIVIDE(archetypesValues, sum)
5:   for all challenges do
6:     interestsPerArchetype ← MULTIPLY(archetypesWeights, challengeArchetypes)
7:     interestPerChallenge ← SUM(interestsPerArchetype)
8:     interestPerChallenge ← interestPerChallenge * 100
9:     playerInterest ← STORE(challenge, interestPerChallenge)
10:   end for
11: end procedure

As shown in Algorithm 2, our algorithm starts by adding 10 to all archetypes' values, changing the
range from [-10, 20] to [0, 30]. Then, it sums all archetypes' values and divides each archetype's value
by this sum, calculating the weight of each archetype (this step is represented by the mathematical
formula 3.1). Afterwards, for each challenge, our algorithm multiplies each archetype's weight by the re-
spective archetype's value of the challenge, resulting in the interest per archetype. Lastly, it sums all the
archetypes' interests, calculating the player's interest per challenge (These last 2 steps are represented
by the mathematical formula 3.2), and multiplies it by 100, making it a percentage.

The challenges and respective player’s interest are stored in a dictionary that has each challenge
object as key and the player’s interest as value. Using the player’s model shown in Listing 3.1 and the
challenge in Listing B.1 as an example, after the aforementioned process the information stored is the
following:

Listing 4.1: Player interest dictionary entry example

<Challenge[0], 13>

where Challenge[0] represents the first challenge in the JSON file, and in this example the only, and has
all the information shown in Listing B.1, and 13 is the player’s interest in that specific challenge (13%).

4.4 Pattern Finder

After the player’s interest stored for each challenge, our algorithm checks which challenges can be
placed and spawned in each walkable cell. First, we read each challenge's pattern, split it in rows and
columns, and store it in a matrix, as illustrated by Fig. 4.1.

Then, we use that pattern's matrix as a “stamp” and apply its center to each walkable cell, comparing
each position of the “stamp” with each cell under it. There is a match in a comparison if there is a “X”

1If no center ("#") is read from the pattern, we assume the center is the first position \((i, j) = (0, 0)\)
in the stamp and the cell under it is non walkable, or if there is any content (or the symbol "#") and
the cell under it is walkable. If the symbol in the “stamp” is “?” there is automatically a match in that
comparison. If there is a complete match\(^2\), this challenge is stored in a list. If a complete match doesn’t
happen, the pattern is rotated counter clockwise and the aforementioned process done once again. Fig.
4.2 shows the “stamp” in a situation with (green) and without (red) a complete match (In this figure, the
pattern’s symbols are almost all evaluated as the symbol “O” and there is only one wall, the last one that
is evaluated as the symbol “X”, as we can see clearer in Fig. 4.1).

This is repeated until there are no more pattern’s rotations available. At the end, the cells and
respective list of challenges that can be placed in those cells are stored in a dictionary, having each cell
as key and the respective list of challenges as value. This step only needs to be done once, being only
necessary to sort the interests in the cells corresponding to the points of the interest curve.

\(^2\)all comparisons return true
4.5 Content Sorting

Having filtered which challenges can be placed in what cells, we finally use the interest curve’s values. Algorithm 3, demonstrates the content sorting process.

\begin{algorithm}
\caption{Content sorting}
\begin{algorithmic}[1]
\Procedure{SortContent}{curveValues, playerInterest}
\ForAll{curveValues}
\State allSqrDifferences $\Leftarrow$ \Call{CalcSqrDifference}{playerInterestValues, curveInterest}
\State allSqrDifferences $\Leftarrow$ \Call{Sort}{allSqrDifferences}
\State challenges $\Leftarrow$ \Call{FindChallenges}{playerInterestChallenges, allSqrDifferences}
\State challengesPerDistance $\Leftarrow$ \Call{Store}{curveDistance, challenges}
\EndFor
\EndProcedure
\end{algorithmic}
\end{algorithm}

So, for each dot in the interest curve, our algorithm calculates the square difference between all player’s interest values and the interest stated by the curve’s dot (this step is represented by the mathematical formula 3.3). The following example illustrates the calculation of a square difference between interests, using the interest from the entry of the playerInterest dictionary (13) in Listing 4.1 and the second value of the second block (65), which represents the interest of the curve’s first dot, present in Listing 3.2:

\[\text{sqrDifference} = (\text{playerInterestValue} - \text{curveInterestValue})^2 = (13 - 65)^2 = 2704\]

In the next step of the process, it sorts all square differences, leaving the lower value on the first position. After, our algorithm finds a challenge corresponding to each one of those sorted differences, resulting in a sorted list of challenges, from the optimal to the least preferred. Lastly, the distance stated by the curve’s dot is stored in a dictionary alongside the respective list of challenges, where the distance is the key and the list of challenges, the value.

4.6 Content Selection

With all content sorted, there is almost no work to be done in the content selection. In this phase, our algorithm will check which cells have the least number of challenges available in their lists, giving these priority, to prevent the appearing of a blank cell or set of cells, when they were supposed to have content placed on them. From the list of cells, we choose one that has the stored distance equal to each curve’s dot’s distance. Having done that, we just have to select the top position in the chosen cell’s challenges’ list and ensure that there are no other challenges of the same type selected. There can only be one challenge of a specific type in the level, so, when a challenge is selected, we block its type from being selected again. This way we can provide diverse challenges to the testers. Each selected cell and respective challenge are stored in a dictionary having the cell as key and challenge as value.
4.7 Content Spawning and Connecting

Having selected the content, we ended up with a dictionary containing only a cell and a challenge per dot in the curve.

Content spawning phase. In this phase, our algorithm reads the pattern from each challenge, and goes through each element. Each pattern’s element has its own identification - doors are identified as Door, monsters as Monster, and so on - and attributes. With this information, our algorithm redirects each element to a specific spawner function. This function is responsible for creating an instance of each element with a specific name, orientation and unique ID, setting each attribute, and adding it to the cell and map.

Content connecting phase. After all elements created and spawned, our algorithm redirects each of them to their respective connector functions. The connector function, as the name suggests, connects the elements creating interactions between them, e.g. a pressure plate that opens a door when pressed.

After content spawned and connected, it is displayed in the interface’s map, as seen in Fig. 4.3, allowing the user to check which challenges were chosen and each element in those challenges. Throughout these two phases, our algorithm stores, dynamically, in runtime, all the placed content’s information to be used in the rewriting of the file that will be interpreted by Legend of Grimrock 2.

![Figure 4.3: Content being displayed](image)
4.8 Dungeon File Rewriting

With all content being spawned and connected, there is only one task missing, exporting the dungeon to the editor. That being said, clicking the Export button calls a function that reads our interface’s map, calling all existing elements’ print functions. These print functions add the elements and all attributes in the correct format to a string, which later is written into the dungeon file. After the rewriting done, a reload command is launched and the editor is forced to refresh, displaying our new dungeon (Fig. 4.4), allowing testers to enjoy our algorithm’s content placement.

![Figure 4.4: New level displayed on editor](image)

4.9 Design Decisions

With this section we want to describe some of the design decisions.

4.9.1 Creating a game vs working on an existing one

This was one of the first decisions made. We had a discussion about the creation of a new game, but we reached a conclusion that it was not feasible, since we needed to validate the game and had to be careful in the content creation. We had to provide diverse content in order to test our hypothesis and that was not possible in the remaining time. So, we opted by working on an existing game, Legend of Grimrock 2, that has diverse content and a mod friendly editor easy to work on.
4.9.2 Creation of an application to help in the challenges’ writing

We had this idea mid implementation and agreed that it would help us in the long run, but, once again, the time remaining was short and we ended up abandoning this idea. This would make the challenge creation a lot easier, once we would have an interface with icons to create challenges instead of writing all challenges in a json file by hand.

4.9.3 Level layout

The level layout we started with was made from zero by us. This layout followed us through almost all implementation. However, we changed the layout before the testing phase. We adapted an existing level made by the creators in order to not impose any type of bias.

4.9.4 Predefined rooms vs anywhere in the map

Our algorithm is able to place content anywhere in the map, but we opted doing 6 predefined rooms, in order to have, sort of, the same layout for each room, because this way we could create and validate challenges for that specific type of room which was not possible if we used all map, since there are numerous patterns and rooms sizes, and that would make it almost impossible to validate distinct challenges for all those different types of room.

4.9.5 Game mechanics

We “deleted”, so to speak, two game mechanics, food and experience. Both mechanics were “deleted” in order not to influence the player’s game experience. Food is a mechanic where if the player doesn’t eat often, he will end up dying. We “deleted” this mechanic, so the player didn’t have to learn one more thing about the game that could be a variable in his experience, which we didn’t want. The other mechanic, experience, was “deleted”, because it wasn’t fair to all players in our test, once it gives an advantage to the players that get challenges with monsters early on - a player with monsters early on will level up, making his characters stronger and more resistant.
## Evaluation

### Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1 Challenge Validation</td>
<td>51</td>
</tr>
<tr>
<td>5.2 Final Evaluation</td>
<td>56</td>
</tr>
<tr>
<td>5.3 Discussion</td>
<td>62</td>
</tr>
</tbody>
</table>
In this chapter we present the evaluations done in our thesis in a more detailed fashion. We describe the challenge validation and our final evaluation, presenting each of these tests' samples, measurements, procedures and results. We end this chapter with a discussion about the obtained results.

5.1 Challenge Validation

Before making players go through the final test, we had to validate the challenges created in order to have a way of arguing about our challenges' identification. We could not simply assume a challenge belonged to a specific archetype. So, after creating the challenges, we made a short video showing each challenge$^{123456}$ and asked players to help us identify each one. The duration of this validation was between 20 and 30 minutes.

5.1.1 Sample

To validate our challenges, we did an exploratory study with 16 participants, from which 12 were male and 4 female, having ages ranging from 12 to 27 years old, with a Mean of 23.06 and a Standard Deviation of 3.699. 25% of the participants said they play video games occasionally when the opportunity presents itself and 75% that they make some time in their schedule to play video games. 31.25% had no professional relation with video games, 50% were students in a course related to video games, where one of them was a MSc student and the remaining 18.75% were researchers in a field related to video games. When asked “Are you familiar with the grid-based dungeon crawler RPG genre”, 50% answered that they play video games but not of this genre and 50% answered that they were familiar with the genre and played at least one game of this genre. Of 16 participants, only 12.5% had ever played a game of the Legend of Grimrock series, while the remaining 87.5% had not.

5.1.2 Measurement

To acquire all the needed information to validate our challenges we used a questionnaire.

**Challenge Validation Questionnaire.** This questionnaire is split in 5 sections. We used the first section of this questionnaire to give some relevant information to the participants, like the reason they were answering this questionnaire and its duration. The second section was used to characterize the participant

$^1$https://youtu.be/2BhNpqfTaYA
$^2$https://youtu.be/KqtO5ldrBU
$^3$https://youtu.be/-gYQOFMSkP8
$^4$https://youtu.be/cVbomyeBipM
$^5$https://youtu.be/Ds5hztVhY84
$^6$https://youtu.be/3oo9lJFzaZM

51
and asks about him and his gaming habits. In the third section, and where we can start gathering relevant information about the challenges, we presented a video related to each challenge and asked the participant to rate, using a 5 points Likert scale, where 1 is “Strongly disagree” and 5 “Strongly agree”, the following statement “I like what this challenge encourages the player to do” after each video. This section was focused on getting information about what the participant thought about each challenge. The fourth section was focused on identifying each challenge as a specific BrainHex archetype. To gather this information, we presented a grid, containing the challenges as columns and sentences that describe each BrainHex archetype being tested as rows, where we asked the participant to match each row to a column, as shown in Appendix D. To make the grid’s columns interpretation clearer, Challenge #1 represents the achiever, Challenge #2 the conqueror, Challenge #3 the daredevil, Challenge #4 the mastermind, Challenge #5 the Seeker and, lastly, Challenge #6 the survivor. The sentences in the grid’s rows are descriptions for each archetype and were written by the creators of BrainHex itself. The correct matches are as follows:

- “You like defeating impossibly difficult foes, struggling until you eventually achieve victory.” - Conqueror
- “You like finding strange and wonderful things, or finding familiar things.” - Seeker
- “You like collecting anything you can collect, and doing everything you possibly can.” - Achiever
- “You like negotiating dizzying platforms or rushing around at high speed while you are still in control.” - Daredevil
- “You like escaping from hideous and scary threats, pulse-pounding risks.” - Survivor
- “You like solving puzzles and devising strategies.” - Mastermind

In the fifth and final section, participants were asked to answer the BrainHex questionnaire and write their results in our questionnaire. This section was used in order to understand if the answers participants gave in the second section were according to their model, mainly their class and subclass.

5.1.3 Procedure

Before asking the participants to start answering the questionnaire, we thanked each one of them for being available to take this test.

We, then, explained that we were creating a game about Legend of Grimrock 2 and testing a set of challenges, and that we would like their help, telling us which they found most interesting.

Having started answering the questionnaire, the participant began by reading the first section, that contained information about why they were answering this questionnaire and its duration. Participant
advanced to the second section where he had demographic questions for participant characterization. Afterwards, on section three, the participant had a video for each challenge followed by the statement “I like what this challenge encourages the player to do.”, where he was asked to rate, using a 5 points Likert scale, where 1 was “Strongly disagree” and 5 “Strongly agree”. Advancing to the fourth section, we presented a grid that contains 6 challenges as column and 6 sentences as rows, where he asked the participant to match each sentence to a challenge. Participant advanced to the fifth section where he had to answer the BrainHex questionnaire and write the results on our questionnaire. We closed this questionnaire with a section where we thanked, once again, the availability and then we bid farewell to the participant.

5.1.4 Results

5.1.4.A BrainHex’s Archetype Identification

With the completion of the challenge validation phase, we were able to gather data about the challenges, how participants identified each challenge and each participant’s BrainHex. Table 5.1 shows the result of our data processing.

<table>
<thead>
<tr>
<th></th>
<th>01</th>
<th>02</th>
<th>03</th>
<th>04</th>
<th>05</th>
<th>06</th>
<th>07</th>
<th>08</th>
<th>09</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Achiever</td>
<td>-0.6</td>
<td>+1.2</td>
<td>-1.7</td>
<td>-0.5</td>
<td>+0.3</td>
<td>+0.7</td>
<td>+1.3</td>
<td>+0.1</td>
<td>+1.7</td>
<td>+2.0</td>
<td>+0.6</td>
<td>-0.1</td>
<td>-0.7</td>
<td>-1.3</td>
<td>+0.3</td>
<td>0.0</td>
</tr>
<tr>
<td>Conqueror</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Daredevil</td>
<td>+0.5</td>
<td>+0.3</td>
<td>+0.3</td>
<td>+1.3</td>
<td>-0.1</td>
<td>+0.9</td>
<td>-0.5</td>
<td>-0.4</td>
<td>+2.1</td>
<td>-0.3</td>
<td>0.0</td>
<td>-1.3</td>
<td>-0.3</td>
<td>+1.3</td>
<td>+0.5</td>
<td>-0.3</td>
</tr>
<tr>
<td>Mastermind</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Seeker</td>
<td>+2.9</td>
<td>+2.1</td>
<td>+2.3</td>
<td>+1.9</td>
<td>+2.3</td>
<td>+1.5</td>
<td>+1.5</td>
<td>+1.5</td>
<td>+1.7</td>
<td>+1.0</td>
<td>+1.0</td>
<td>-1.3</td>
<td>+0.3</td>
<td>-0.1</td>
<td>+1.3</td>
<td>+0.7</td>
</tr>
<tr>
<td>Survivor</td>
<td>+1.5</td>
<td>+1.3</td>
<td>-0.2</td>
<td>+0.1</td>
<td>+0.1</td>
<td>+1.9</td>
<td>-0.2</td>
<td>+0.5</td>
<td>-0.6</td>
<td>-1.3</td>
<td>-2.1</td>
<td>-0.9</td>
<td>+0.3</td>
<td>-0.1</td>
<td>+1.3</td>
<td>-0.1</td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Table 5.1: Challenge validation’s processed data

In this table, each row represents a BrainHex archetype and each column, a participant. For each archetype, we divide it in 2 rows. The first row shows the alignment between the value of that BrainHex archetype and the video for the challenge that belongs to that archetype. To get these results, we applied the following mathematical formula:

\[
\text{result} = \text{video} - \left(\frac{\text{BrainHex} + 10}{7.5}\right) + 1
\]  

(5.1)

where \(\text{result}\) is a value in the range \([-4, 4]\), \(\text{video}\) is the participant’s video rating, \([1, 5]\), of a specific
archetype, BrainHex is the participant’s BrainHex value, [-10, 20], for that archetype, and \(((BrainHex + 10)/7.5) + 1\) is the normalization of the BrainHex value, changing its range from [-10, 20] to [1, 5]. With the results of this subtraction, we can verify if the participants rated the videos above (positive) or below (negative) of what their BrainHex values suggested, making it easier to check the alignment between the BrainHex profile and the challenges’ assessment by participants. The second row presents all errors done by the participants and in which challenges these occurred. Is considered an error when the participant fails to match a sentence to the respective challenge.

This phase’s main goal was to identify each challenge as a BrainHex archetype. As seen in Table 5.1, 62.5\% (10) of the participants were able to identify every challenge as the correct BrainHex archetype, once there were no errors in the grid section of the questionnaire. We can also verify that 31.25\% (5) had 2 errors and 6.25\% (1) had 3, resulting in 13 errors out of 96 answers, which results in, approximately, 13.5\% of the answers being errors. This also means that there were, approximately, 86.5\% of correct answers. Analyzing the answers for each archetype, we can verify that:

- Achiever: there were 2 errors, so, this means that there were 87.5\% of correct matches.
- Conqueror: there was only 1 error, which means that 93.75\% of the answers were correct.
- Daredevil: the same as in Achiever, 87.5\% of correct matches.
- Mastermind: 87.5\% of correct answers, as in Achiever and Daredevil.
- Seeker: there were 5 errors in 16 answers, so 31.25\% of errors. However, there were more than the double of correct answers, 11, which means that 68.75\% were correct answers.
- Survivor: as in Conqueror, there were 93.75\% of correct matches.

Although seeker had 5 errors, if we analyze the errors of each participant, present in Table 5.1, we can verify that there was an error between seeker and daredevil, seeker and conqueror and even seeker and survivor. Knowing that the seeker challenge is, basically, a hidden button to open a hidden door that leads to a floating power gem and the sentences that describe each of the challenges in the errors are: “You like negotiating dizzying platforms or rushing around at high speed while you are in control” (daredevil), “You like defeating impossibly difficult foes, struggling until you eventually achieve victory” (conqueror) and “You like escaping from hideous and scary threats, pulse-pounding risks” (survivor), we assume that these errors were due to lack of attention when reading the sentences or matching sentences to challenges. However, there were credible matching errors between achiever and seeker, since picking up and storing the power gem can be interpreted as “You like collecting anything you can collect, and doing everything you possible can”, which is the sentence used to describe the achiever archetype.
So, with all the aforementioned information, we conclude that the designed challenges were understood as intended.

5.1.4.B BrainHex Profile - Challenges’ Assessment Alignment

Having analyzed the errors, let’s look at the other information acquired in the questionnaire, the alignment between the BrainHex profile and the challenges’ assessment by participants. So, if the result’s value for a specific archetype is 0.0, it means that a participant had a perfect alignment between the video rating and the BrainHex profile, for that specific archetype. However, a perfect alignment is rare, so, instead of expecting a result of 0.0, we verified that there is an alignment less or equal than 1.0 in 53 of 96 answers, which means that there is an alignment less or equal than 1.0 between the BrainHex profile and the challenges’ assessment by participants in, approximately, 55.21% of the cases. Although there is an alignment less or equal than 1.0 in, approximately, 55.21% of the cases, there are still some relevant deviations. Having noticed these deviations, we contacted some of the participants, trying to obtain more information in order to improve the alignment between the BrainHex profile and the challenges’ assessment by participants. One participant told us that he gave a higher value to a challenge he didn’t like, because it was not a good challenge for him, but it could be for someone else. We had to check if this was happening to any other participants, so, we asked four participants, where we noticed some deviations, to rate once more the videos, but, this time, we wanted two distinct ratings, the perception of the interest (1) for them personally and (2) for the community of players in general. Table 5.2 shows this new data.

<table>
<thead>
<tr>
<th></th>
<th>Achiever</th>
<th>Conqueror</th>
<th>Daredevil</th>
<th>Mastermind</th>
<th>Seeker</th>
<th>Survivor</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>2 3 3 4 3 5 5 4 5 4 5 4 3 3 2 4 2 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>02</td>
<td>4 4 4 4 2 5 4 2 4 5 5 5 5 4 4 2 2 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>03</td>
<td>5 3 5 4 4 4 2 3 2 3 4 3 4 4 3 4 3 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>04</td>
<td>4 4 4 5 5 5 4 4 4 2 4 2 1 2 3 4 3 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5.2: Second iteration with 4 participants

In this table, each row represents one of the four participants and each column, one BrainHex archetype. Each archetype’s column is divided in 3: original, “personal” and “general” ratings. With this second iteration, we were able to verify that from the 24 new answers:

- 12 “general” answers are closer to the originals.
- 4 “personal” answers are closer to the originals.

---

7each pair personal, general counts as 1 answer
• 8 do not distinguish between “personal” and “general”.

With this information, we verified that the deviated results, in general, may have been due to a misinterpretation of the question in the questionnaire.

5.2 Final Evaluation

Having the challenges validated and implementation completed, we were ready for the final tests, where we would try to verify our hypothesis. To test our model, we asked some players to play a level with content placed taking their model into account and other players to play a level with content placed taking the inverse of their model into account. With this we were able to compare the results from both levels and see which provided a greater game experience to the player. This final test had the duration of 30-45 minutes.

5.2.1 Sample

To test our model's content placement we had 30 participants playing a level of Legend of Grimrock 2. These 30 participants were divided into 2 groups: the players from one group played the level with content placed taking their model into account and the players from the other played a level with content placed taking the inverse of their model into account. From these 30 participants, 20 were male and 10 female, having ages ranging from 19 to 27 years old, with a Mean of 23.13 and a Standard Deviation of 1.928. 63.3% of the participants said they make some time in their schedule to play video games, 33.3% that they play video games occasionally when the opportunity presents itself and 3.3% that they didn’t play video games. 66.6% had no professional relation with video games, 26.6% were students in a course related to video games and the remaining 6.6% were researchers in a field related to video games. When asked “Are you familiar with the grid-based dungeon crawler RPG genre”, 63.3% answered that they play video games but not of this genre, 30% answered that they were familiar with the genre and played at least one game of this genre, 3.3% answered that this genre was one of their favorites, and they played several games of this genre and the remaining 3.3% answered that they didn’t play video games. Of 30 participants, only 6.6% had ever played a game of the Legend of Grimrock series, while the remaining 93.3% had not.

5.2.2 Measurements

To gather all the needed data in order to try to support our hypothesis and test our model’s content placement, we used a questionnaire and a short interview.
**Participant’s Opinion Questionnaire.** The questionnaire we created is divided in 3 sections. The first section was used to give relevant information about the test they were doing, like what were we doing and why they were answering this and the questionnaire’s duration. Second section was, as our first questionnaire, to characterize the participants. In this section we ask about age, sex and gaming habits. The last section, and the one from where we obtain the data we need, is the core module from the game experience questionnaire [30]. This section was focused on probing the participant’s feelings and thoughts while playing the level. We didn’t have a section for the BrainHex in this questionnaire, because we have the BrainHex questionnaire in our application. The full version of the participant’s opinion questionnaire is annexed in Appendix E.

**Short Interview.** This interview only happened if the participants accepted to play again. In case they wanted to play again, it would be given the opposite version and the information would not be gathered by the questionnaire but the interview instead. This interview only contains 2 questions:

- What is the preferred sequence of challenges?
- Why?

This interview was focused on retrieving any type of anecdotal information, in order to confirm that participants who played both versions - model and inverted model - actually had preference for a particular ordering of the same set of challenges.

### 5.2.3 Procedure

As in the last test’s procedure, we started by thanking each one of participants for their availability to take this test.

Next, we explained that we were creating a game about Legend of Grimrock 2 and testing content for a level, and that we would like their help, giving us their opinion. We also explained that we were asking for their player model so that we could take into account what the participants value most in a game.

Before starting the final test, we presented a tutorial level to the participants in order for them to try the game’s controls and mechanics.

After reaching the end of the tutorial level, we launched our application and changed the mode to testing. The participants received the application in testing mode, where they cannot see the challenges, map or curve, and were asked to answer the BrainHex questionnaire.

After answering the questionnaire, we saved the participants’ models, loaded the components we needed and made our algorithm run, which generates an ID for each participant in order to maintain anonymity. Then, we exported the results to the level, which was already open in the editor. The participant played the exported level in the editor’s full screen preview.
Having reached the end of the level, we asked the participants to answer the final questionnaire, where they started by inserting their ID and answering the demographic section. Afterwards, the participants answered to the next section of the questionnaire, referring to the core module of the game experience questionnaire [30]. We closed this questionnaire with a section where we thanked, once again, the availability of the participant.

Then, we asked the participants if they wanted to play again. When we got affirmative answers, we inverted their model, loaded the remaining components, ran the algorithm and exported the new level. We ended this point by doing a short interview where we asked which was the preferred sequence and why.

Finally, we thanked one more time and bid farewell to the participants.

5.2.4 Results

With the final tests phase completed, we gathered data from 30 questionnaires regarding the evaluation of our model’s content placement. Once we had the core module of the game experience questionnaire [30] as part of our test, we used their scoring guidelines. The core module of the game experience questionnaire [30] assesses game experience as scores on seven components: competence, sensory and imaginative immersion, flow, tension/annoyance, challenge, negative affect and positive affect.

Fig. 5.1 shows how participants scored in the competence component.

![Figure 5.1: Competence component](image)

Competence, and the rest of the components, is divided in two participant’s opinion #1 and #2. Participant’s opinion #1 refers to content placement taking their model into account and participant’s opinion #2 to content placement taking the inversion of their model into account. From here onward, we use #1 and #2 when referring to participant’s opinion #1 and participant’s opinion #2, respectively. So, while explaining the Competence results, we use Competence #1 and #2 to distinguish between both opinions. Competence #1 has a Range of 3.00, having a Minimum value of 0.40 and a Maximum of
3.40. Its Mean and Standard Deviation values are 2.15 and 0.86, respectively. Competence #2 has the same Range, Minimum and Maximum values. However, its Mean and Standard Deviation values are 1.97 and 1.01, respectively. After doing a Mann-Whitney test, we verified that Competence #1 had a Mean Rank of 16.17, Competence #2 had a Mean Rank of 14.83, \( Z = -0.417 \) and \( p = 0.676 \). With this data, we fail to reject the null hypothesis\(^8\). Given that the challenges are the same but presented in a different order, the overall competence is expected to be no different.

In Fig. 5.2 we present how participants scored in sensory and imaginative immersion.

![Figure 5.2: Sensory and imaginative immersion component](image)

Sensory and imaginative immersion #1 has a Range of 3.17, with Minimum and Maximum values of 0.00 and 3.17, respectively. Its Mean value is 1.87 and Standard Deviation value is 0.65. Sensory and imaginative immersion #2 has a Range of 2.67, having a Minimum value of 0.33 and Maximum value of 3.00. Its Mean and Standard Deviation values are 1.99 and 0.80, respectively. Having done the Mann-Whitney test, we obtained the following results. Mean Ranks of 14.20 and 16.80 to Sensory and imaginative immersion #1 and Sensory and imaginative immersion #2, respectively, \( Z = -0.815 \) and \( p = 0.415 \), which makes us fail to reject the null hypothesis.

Next, we show in Fig. 5.3 how participants scored in the flow component.

Flow #1’s Minimum and Maximum values are 0.20 and 4.00, respectively, making a Range of 3.80. Its Mean and Standard Deviation values are 2.13 and 0.99, respectively. On the other hand, Flow #2 has a Range of 3.20, with a Minimum value of 0.60 and Maximum value of 3.80. The Mean and Standard Deviation values are 2.35 and 1.07, respectively. The Mean Ranks attributed to Flow #1 and Flow #2 after the Mann-Whitney test are 14.70 and 16.30, respectively. In addition, with \( Z = -0.500 \) and \( p = 0.617 \), we fail to reject the null hypothesis.

Fig. 5.4 displays how participants scored in the tension/annoyance component.

Tension #1 has a Range of 2.00, having a Minimum value of 0.00 and a Maximum of 2.00. Its Mean

\(^8\)there is no difference between the ranks of the two Competences
and Standard Deviation values are 0.62 and 0.67, respectively. Tension #2 has the double of Tension #1’s Range (4.00), with Minimum and Maximum values of 0.00 and 4.00, respectively. Its Mean and Standard Deviation values are 1.27 and 1.16, respectively. After doing a Mann-Whitney test, we verified that Tension #1 had a Mean Rank of 12.97, Tension #2 had a Mean Rank of 18.03, $Z = -1.604$ and $p = 0.109$. With this data, we do not have enough evidence to conclude that the difference between Tension #1 and Tension #2 is statistically significant. So, we fail to reject the null hypothesis. However, there seems to be a tendency to the increase of Tension #2, which may become statistically significant with more participants.

In Fig. 5.5 we present the participants’ scores in the challenge component.

In Challenge #1, we have Minimum and Maximum values of 0.00 and 2.60, respectively, which creates a Range of 2.60. The Mean and Standard Deviation values are 1.45 and 0.72, respectively. In Challenge #2, we have a Range of 3.00, having a Minimum value of 0.60 and a Maximum value of 3.60. Its Mean and Standard Deviation values are 1.68 and 0.80, respectively. Using the Mann-Whitney
test, we fail to reject the null hypothesis, since the Mean Ranks attributed to Challenge #1 and Challenge #2 are 14.70 and 16.30, respectively, $Z = -0.500$ and $p = 0.617$. Given that the challenges are the same but presented in a different order, the overall challenge is expected to be no different.

Fig. 5.6 shows how participants scored in the negative affect component.

Negative affect #1 has a Range of 2.25, having a Minimum value of 0.00 and a Maximum value of 2.25. It has a Mean value of 0.77 and a Standard Deviation value of 0.60. In Negative affect #2, the Minimum and Maximum values are 0.00 and 2.00, respectively, making a Range of 2.00. Its Mean and Standard Deviation values are 0.77 and 0.66, respectively. With the data obtained from Mann-Whitney - Negative affect #1’s Mean Rank is 15.70, Negative affect #2’s Mean Rank is 15.30, $Z = -0.126$ and $p = 0.900$ - we do not have enough evidence to conclude that the difference between Negative affect #1 and Negative affect #2 is statistically significantly. So, we fail to reject the null hypothesis.

Lastly, Fig. 5.7 presents how participants scored in the positive affect component.

On one hand, Positive affect #1 has a Range of 3.00, with Minimum and Maximum values of 0.80
and 3.80, respectively. Its Mean and Standard Deviation are 2.96 and 0.71, respectively. On the other hand, Positive affect #2 has a Range of 2.80, with Minimum and Maximum values of 0.80 and 3.60, respectively. Its Mean and Standard Deviation are 2.33 and 0.78, respectively. By doing the Mann-Whitney test, we verified that Positive affect #1’s Mean Rank is 19.27, Positive affect #2’s Mean Rank is 11.73, $Z = -2.363$ and $p = 0.018$. With this data, we reject the null hypothesis and have enough evidence to conclude that the difference between Positive affect #1 and Positive affect #2 is statistically significantly.

In table 5.3, we present a summary of all the results previously discussed.

We were also able to obtain anecdotal information from 3 participants that played both levels. All 3 said that they preferred level A - where content was placed taking their model into account - instead of B - where content was placed taking the inverse of their model into account. When asked "Why?" the answers were the following:

- “I preferred A, because in B I got more things that I didn’t like in a row.”
- “I preferred A, because I got what I liked the most in the beginning and what I liked the least interspersed with other challenges that I liked.”
- “I preferred A, because B started with something I hated.”

5.3 Discussion

In the Challenge Validation, we got enough information to conclude that the designed challenges were understood as intended. 62.5% of the participants were able to identify all these challenges as the correct BrainHex archetype. Of the remaining participants - that got 2 and 3 errors - we withdrew their feedback in order to improve each challenge. These results were in the range of results we were
Table 5.3: Summary of the final evaluation results

<table>
<thead>
<tr>
<th>Component</th>
<th>Range</th>
<th>Min</th>
<th>Max</th>
<th>Mean</th>
<th>Std Dev</th>
<th>Mean Rank</th>
<th>Z</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>Competence</td>
<td>3.00</td>
<td>0.40</td>
<td>3.40</td>
<td>2.15</td>
<td>0.86</td>
<td>16.17</td>
<td>-0.417</td>
<td>0.676</td>
</tr>
<tr>
<td>Immersion</td>
<td>3.17</td>
<td>0.00</td>
<td>3.17</td>
<td>1.87</td>
<td>0.65</td>
<td>14.20</td>
<td>-0.815</td>
<td>0.415</td>
</tr>
<tr>
<td>Flow</td>
<td>3.80</td>
<td>0.20</td>
<td>4.00</td>
<td>2.13</td>
<td>0.99</td>
<td>14.70</td>
<td>-0.500</td>
<td>0.617</td>
</tr>
<tr>
<td>Tension</td>
<td>2.00</td>
<td>0.00</td>
<td>2.00</td>
<td>0.62</td>
<td>0.67</td>
<td>12.97</td>
<td>-1.604</td>
<td>0.109</td>
</tr>
<tr>
<td>Challenge</td>
<td>2.60</td>
<td>0.00</td>
<td>2.60</td>
<td>1.45</td>
<td>0.72</td>
<td>14.70</td>
<td>-0.500</td>
<td>0.617</td>
</tr>
<tr>
<td>Negative</td>
<td>2.25</td>
<td>0.00</td>
<td>2.25</td>
<td>0.77</td>
<td>0.60</td>
<td>15.70</td>
<td>-0.126</td>
<td>0.900</td>
</tr>
<tr>
<td>Positive</td>
<td>3.00</td>
<td>0.80</td>
<td>3.80</td>
<td>2.96</td>
<td>0.71</td>
<td>19.27</td>
<td>-2.363</td>
<td>0.018</td>
</tr>
</tbody>
</table>

expecting. We did several iterations on those challenges, until there were no more ideas surging for modifications. We asked for third opinions and people to associate each sentence to a challenge while doing those iterations and the results were positive.

We also got several information about the alignment between BrainHex profiles and videos. Although we verified that there was an alignment in 53 of 96 answers, which means that there was an alignment between the BrainHex profile and the challenges’ assessment by participants in, approximately, 55.21% of the cases, we were worried about the existing deviations at some point, because some of them were significant. By talking to the participants, we verified that the deviated results may have been due to a misinterpretation of the question in the questionnaire.

With the help of the core module of the game experience questionnaire [30], we were able to gather relevant information about our model’s content placement, into the form of 7 distinct components - competence, sensory and imaginative immersion, flow, tension/annoyance, challenge, negative affect and positive affect. With the results obtained from 30 questionnaires, we got an interesting result. We were able to verify that there is only 1 component that is affected by our model’s content placement, positive affect. The Mean Rank from positive affect in Participant’s opinion #1 is 19.27, which is almost the double of the one in Participant’s opinion #2. Moreover, it got $Z = -2.363$ and $p = 0.018$. These values seem to indicate that the ordering of challenges according to the curve of interest appropriate to the
player’s model only affects the positive emotions reported by the player, but does so in a very clear way. That is, choosing the moment when the challenges most valued by the player appear had no impact on the less positive aspects of the experience but amplified the positive moments.
6

Conclusion

Contents

6.1 Future Work .................................................. 67
With this thesis we intended to discover if the order of the content's presentation had an effect in the player's game experience. We wanted to verify if using an interest curve to place and pace content based on its relevance to the player's interest would help creating a better game experience for the player.

We began by searching for works related to what we want to do. We searched for personality models, player models, procedural content generation and progression, presenting each of the found works. In personality models, we described the Myers-Briggs Type Indicator [5], Five Factor Model [9] and Cloninger’s Temperament and Character Inventory [18]. In players models, we presented Bartle Player Types [21], Demographic Game Design [22], BrainHex [23] and Quantic Foundry’s Gamer Motivation Model [24]. In procedural content generation, we described Experience-Driven Procedural Content Generation [26] and 3Buddy [27]. Lastly, in progression, we presented Facade [29]. After presenting and describing each work, we reflected in which works would be beneficial for us to use as inspiration.

Using some of the related work as inspiration, we designed our computational model that receives 4 components as parameters: player’s model, interest curve, challenge library and level. Throughout a run, our model calculates game distances, rates and ranks challenges by player interest, reads the challenges’ patterns, trying to find each one of these in the map, selects content by comparing how relevant it is for the player and the curve’s interest we want to match and, finally, places the selected content. In addition to this implementation, we created an interface that allows a better interaction with the model.

To test our model, we first validated the challenges we were going to use. Then, we asked people to play a level with this model’s content placement - 15 with the content placed taking their model into account and 15 with the content placed taking the inverse of their model into account - having them answer a questionnaire and an interview in the end. With this test, we obtained and analyzed data from 7 distinct components of the game experience questionnaire’s core module [30].

This data gathered from the questionnaires supports our hypothesis, since it suggests that our model’s content placement affected the positive affect, where the higher values were present in the content placed taking the player’s model into account, so, the results indicate that our model’s content placement was able to provide a better game experience to the player.

6.1 Future Work

Although our model’s results were very interesting, we would like to see it improve. Throughout our model’s development, we had some ideas that we think are worth exploring, since they may produce even more interesting results.

- Application to create challenges. Throughout the challenges’ creation we had to open the editor,
place each piece of content where we wanted them, make the connections, add items and then write all those in a json file. With the creation of this application, we would be able to select the grid size for the challenge, which challenges go into what cells, the connections, the items, and, more importantly, export the result directly to a json file.

- **Try different interest curves.** We used an adaptation of the story arc, but it would be interesting to see content being placed using several interest curves and analyze which one produced the best results.

- **Use BrainHex's exceptions.** Our only content selector is the curve. We would like to see the challenges adapted, or restrictions added, in order to accommodate BrainHex's exceptions. It could be interesting to see how the content placement would compare to our model's.

- **Use different player models.** At the moment, the player model being used is the BrainHex. However, using a different player model could bring new challenges and restrictions to the table.

- **Expand the challenge library.** Our initial idea was to have 70 different challenges in the library. However, we couldn’t achieve that in the time we had. Creating new challenges seems like an interesting work. This would make it possible to place content wherever we want in the level, instead of placing it in 6 predefined rooms. This would also make it possible to create challenges that are identified as more than one archetype at the same time.

- **Use this model in another game.** Adapt our model to another game. Would the results still be the same if our model was used in a completely different game? This could be an interesting discovery.
Bibliography


[27] P. Lucas and C. Martinho, “Stay awhile and listen to 3buddy, a co-creative level design support tool,” in Eighth International Conference on Computational Creativity, ICCC, Atlanta, 2017.


This appendix shows the code of the dungeon used in our thesis.

Listing A.1: Legend of Grimrock 2 dungeon code in Lua

```
-- This file has been generated by Dungeon Editor 2.2.4

--- level 1 ---

newMap{
    name = "Unnamed",
    width = 32,
    height = 32,
    levelCoord = {0,0,0},
    ambientTrack = "dungeon",
    tiles = {
        ...)
"dungeon_floor",
"dungeon_wall",
}
}
loadLayer("tiles", {
2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,
2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,
2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,
2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,
2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,
2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,
2,2,2,2,2,1,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,
2,2,2,2,1,2,2,2,2,1,2,1,1,1,1,1,1,1,1,2,1,2,1,1,1,2,1,2,1,2,2,2,
2,2,2,2,1,2,2,2,2,1,2,2,2,1,2,2,2,1,2,2,2,2,2,2,2,2,2,2,2,2,2,
2,2,2,2,1,2,1,1,1,1,1,1,1,1,1,2,1,2,1,2,1,2,1,2,1,2,1,2,2,2,2,
2,2,2,2,1,2,1,1,1,1,1,1,1,1,2,1,2,1,2,1,2,1,2,1,2,1,2,2,2,2,2,
2,2,2,2,1,2,1,1,1,1,1,1,1,1,2,1,2,1,2,1,2,1,2,1,2,1,2,2,2,2,2,
2,2,2,2,1,2,1,1,1,1,1,1,1,2,1,2,1,2,1,2,1,2,1,2,1,2,2,2,2,2,2,
2,2,2,2,1,2,1,1,1,1,1,1,1,2,1,2,1,2,1,2,1,2,1,2,1,2,2,2,2,2,2,
2,2,2,2,1,2,1,1,1,1,1,1,1,2,1,2,1,2,1,2,1,2,1,2,1,2,2,2,2,2,2,
2,2,2,2,1,2,1,1,1,1,1,1,1,2,1,2,1,2,1,2,1,2,1,2,1,2,2,2,2,2,2,
2,2,2,2,1,2,1,1,1,1,1,1,1,2,1,2,1,2,1,2,1,2,1,2,1,2,2,2,2,2,2,
2,2,2,2,1,2,1,1,1,1,1,1,1,2,1,2,1,2,1,2,1,2,1,2,1,2,2,2,2,2,2,
2,2,2,2,1,2,1,1,1,1,1,1,1,2,1,2,1,2,1,2,1,2,1,2,1,2,2,2,2,2,2,
2,2,2,2,1,2,1,1,1,1,1,1,1,2,1,2,1,2,1,2,1,2,1,2,1,2,2,2,2,2,2,
2,2,2,2,1,2,1,1,1,1,1,1,1,2,1,2,1,2,1,2,1,2,1,2,1,2,2,2,2,2,2,
2,2,2,2,1,2,1,1,1,1,1,1,1,2,1,2,1,2,1,2,1,2,1,2,1,2,2,2,2,2,2,
2,2,2,2,1,2,1,1,1,1,1,1,1,2,1,2,1,2,1,2,1,2,1,2,1,2,2,2,2,2,2,
2,2,2,2,1,2,1,1,1,1,1,1,1,2,1,2,1,2,1,2,1,2,1,2,1,2,2,2,2,2,2,
2,2,2,2,1,2,1,1,1,1,1,1,1,2,1,2,1,2,1,2,1,2,1,2,1,2,2,2,2,2,2,
spawn ("torch_holder",5,19,0,0,"torch_holder_1")
torch_holder_1.controller : setHasTorch (true)
spawn ("torch_holder",8,22,1,0,"torch_holder_2")
torch_holder_2.controller : setHasTorch (true)
spawn ("torch_holder",7,21,3,0,"torch_holder_3")
torch_holder_3.controller : setHasTorch (true)
spawn ("torch_holder",12,19,0,0,"torch_holder_4")
torch_holder_4.controller : setHasTorch (true)
spawn ("torch_holder",11,22,2,0,"torch_holder_5")
torch_holder_5.controller : setHasTorch (true)
spawn ("torch_holder",12,26,2,0,"torch_holder_6")
torch_holder_6.controller : setHasTorch (true)
spawn ("torch_holder",16,25,1,0,"torch_holder_7")
torch_holder_7.controller : setHasTorch (true)
spawn ("torch_holder",16,22,0,0,"torch_holder_8")
torch_holder_8.controller : setHasTorch (true)
spawn ("torch_holder",15,22,2,0,"torch_holder_9")
torch_holder_9.controller : setHasTorch (true)
spawn ("torch_holder",15,17,0,0,"torch_holder_10")
torch_holder_10.controller : setHasTorch (true)
spawn ("torch_holder",10,17,3,0,"torch_holder_11")
torch_holder_11.controller : setHasTorch (true)
spawn ("torch_holder",12,17,2,0,"torch_holder_12")
torch_holder_12.controller : setHasTorch (true)
spawn ("torch_holder",9,16,2,0,"torch_holder_13")
torch_holder_13.controller : setHasTorch (true)
spawn ("torch_holder",9,14,0,0,"torch_holder_14")
torch_holder_14.controller : setHasTorch (true)
spawn ("torch_holder",7,12,0,0,"torch_holder_15")
torch_holder_15.controller : setHasTorch (true)
spawn("torch_holder",6,15,2,0,"torch_holder_16")
torch_holder_16.controller:setHasTorch(true)
spawn("torch_holder",13,12,0,0,"torch_holder_17")
torch_holder_17.controller:setHasTorch(true)
spawn("torch_holder",18,12,0,0,"torch_holder_18")
torch_holder_18.controller:setHasTorch(true)
spawn("torch_holder",18,15,2,0,"torch_holder_19")
torch_holder_19.controller:setHasTorch(true)
spawn("torch_holder",19,19,1,0,"torch_holder_20")
torch_holder_20.controller:setHasTorch(true)
spawn("torch_holder",19,25,2,0,"torch_holder_21")
torch_holder_21.controller:setHasTorch(true)
spawn("torch_holder",24,25,1,0,"torch_holder_22")
torch_holder_22.controller:setHasTorch(true)
spawn("torch_holder",22,23,0,0,"torch_holder_23")
torch_holder_23.controller:setHasTorch(true)
spawn("torch_holder",22,21,2,0,"torch_holder_24")
torch_holder_24.controller:setHasTorch(true)
spawn("torch_holder",21,19,3,0,"torch_holder_25")
torch_holder_25.controller:setHasTorch(true)
spawn("torch_holder",23,18,0,0,"torch_holder_27")
torch_holder_27.controller:setHasTorch(true)
spawn("torch_holder",24,13,0,0,"torch_holder_26")
torch_holder_26.controller:setHasTorch(true)
spawn("torch_holder",26,13,1,0,"torch_holder_28")
torch_holder_28.controller:setHasTorch(true)
spawn("torch_holder",26,5,0,0,"torch_holder_29")
torch_holder_29.controller:setHasTorch(true)
spawn("torch_holder",26,9,1,0,"torch_holder_30")
torch_holder_30.controller:setHasTorch(true)
spawn("torch_holder",22,6,2,0,"torch_holder_31")
torch_holder_31.controller:setHasTorch(true)
spawn("torch_holder",23,11,2,0,"torch_holder_32")
torch_holder_32.controller:setHasTorch(true)
spawn("torch_holder",18,10,3,0,"torch_holder_33")
torch_holder_33.controller:setHasTorch(true)
spawn("torch_holder",18,5,0,0,"torch_holder_34")
torch_holder_34.controller:setHasTorch(true)
spawn("torch_holder",16,5,3,0,"torch_holder_35")
torch_holder_35.controller:setHasTorch(true)
spawn("torch_holder",16,6,1,0,"torch_holder_36")
torch_holder_36.controller:setHasTorch(true)
spawn("torch_holder",14,6,0,0,"torch_holder_38")
torch_holder_38.controller:setHasTorch(true)
spawn("torch_holder",10,6,0,0,"torch_holder_37")
torch_holder_37.controller:setHasTorch(true)
spawn("torch_holder",12,9,2,0,"torch_holder_39")
torch_holder_39.controller:setHasTorch(true)
spawn("torch_holder",6,8,1,0,"torch_holder_41")
torch_holder_41.controller:setHasTorch(true)
spawn("starting_location",5,19,2,0,"starting_location_1")
spawn("dungeon_stairs_up",5,5,0,0,"dungeon_stairs_up_1")
spawn("chest",5,21,0,0,"chest_1")
spawn("cutlass",5,21,0,0,"cutlass_1")
chest_1.surface:addItem(cutlass_1.item)
spawn("skullcleave",5,21,0,0,"skullcleave_1")
chest_1.surface:addItem(skullcleave_1.item)
spawn("crossbow",5,21,0,0,"crossbow_1")
chest_1.surface:addItem(crossbow_1.item)
spawn("quarrel",5,21,0,0,"quarrel_1")
quarrel_1.item:setStackSize(100)
chest_1.surface:addItem(quarrel_1.item)
spawn("lightning_rod",5,21,0,0,"lightning_rod_1")
chest_1.surface:addItem(lightning_rod_1.item)
spawn("leather_gloves",5,21,0,0,"leather_gloves_1")
chest_1.surface:addItem(leather_gloves_1.item)
spawn("leather_gloves",5,21,0,0,"leather_gloves_2")
chest_1.surface:addItem(leather_gloves_2.item)
spawn("tattered_cloak",5,21,0,0,"tattered_cloak_1")
chest_1.surface:addItem(tattered_cloak_1.item)
```javascript
spawn("peasant_cap",5,21,0,0,"peasant_cap_1")
chest_1.surface:addItem(peasant_cap_1.item)
spawn("peasant_cap",5,21,0,0,"peasant_cap_2")
chest_1.surface:addItem(peasant_cap_2.item)
spawn("tattered_shirt",5,21,0,0,"tattered_shirt_1")
chest_1.surface:addItem(tattered_shirt_1.item)
spawn("tattered_shirt",5,21,0,0,"tattered_shirt_2")
chest_1.surface:addItem(tattered_shirt_2.item)
spawn("leather_boots",5,21,0,0,"leather_boots_1")
chest_1.surface:addItem(leather_boots_1.item)
spawn("leather_boots",5,21,0,0,"leather_boots_2")
chest_1.surface:addItem(leather_boots_2.item)
spawn("leather_pants",5,21,0,0,"leather_pants_1")
chest_1.surface:addItem(leather_pants_1.item)
spawn("leather_pants",5,21,0,0,"leather_pants_2")
chest_1.surface:addItem(leather_pants_2.item)
spawn("potion_healing",5,21,0,0,"potion_healing_1")
potion_healing_1.item:setStackSize(8)
chest_1.surface:addItem(potion_healing_1.item)
spawn("potion_energy",5,21,0,0,"potion_energy_1")
potion_energy_1.item:setStackSize(8)
chest_1.surface:addItem(potion_energy_1.item)
spawn("tattered_cloak",5,21,0,0,"tattered_cloak_2")
chest_1.surface:addItem(tattered_cloak_2.item)
spawn("torch_holder",9,23,2,0,"torch_holder_43")
torch_holder_43.controller:setHasTorch(true)
spawn("torch_holder",9,21,0,0,"torch_holder_46")
torch_holder_46.controller:setHasTorch(true)
spawn("torch_holder",12,24,1,0,"torch_holder_47")
torch_holder_47.controller:setHasTorch(true)
spawn("torch_holder",14,25,0,0,"torch_holder_48")
torch_holder_48.controller:setHasTorch(true)
spawn("torch_holder",15,20,1,0,"torch_holder_49")
torch_holder_49.controller:setHasTorch(true)
spawn("torch_holder",15,19,3,0,"torch_holder_50")
```
torch_holder_50.controller:setHasTorch(true)
spawn("torch_holder",10,15,1,0,"torch_holder_51")
torch_holder_51.controller:setHasTorch(true)
spawn("torch_holder",10,12,2,0,"torch_holder_52")
torch_holder_52.controller:setHasTorch(true)
spawn("torch_holder",15,14,2,0,"torch_holder_53")
torch_holder_53.controller:setHasTorch(true)
spawn("torch_holder",15,12,0,0,"torch_holder_56")
torch_holder_56.controller:setHasTorch(true)
spawn("torch_holder",19,17,3,0,"torch_holder_57")
torch_holder_57.controller:setHasTorch(true)
spawn("torch_holder",19,21,3,0,"torch_holder_58")
torch_holder_58.controller:setHasTorch(true)
spawn("torch_holder",19,23,1,0,"torch_holder_59")
torch_holder_59.controller:setHasTorch(true)
spawn("torch_holder",21,25,0,0,"torch_holder_60")
torch_holder_60.controller:setHasTorch(true)
spawn("torch_holder",21,22,3,0,"torch_holder_61")
torch_holder_61.controller:setHasTorch(true)
spawn("torch_holder",23,22,1,0,"torch_holder_62")
torch_holder_62.controller:setHasTorch(true)
spawn("torch_holder",14,13,3,0,"torch_holder_44")
torch_holder_44.controller:setHasTorch(true)
spawn("torch_holder",24,16,3,0,"torch_holder_45")
torch_holder_45.controller:setHasTorch(true)
spawn("torch_holder",24,14,1,0,"torch_holder_54")
torch_holder_54.controller:setHasTorch(true)
spawn("torch_holder",26,11,3,0,"torch_holder_55")
torch_holder_55.controller:setHasTorch(true)
spawn("torch_holder",26,7,3,0,"torch_holder_63")
torch_holder_63.controller:setHasTorch(true)
spawn("torch_holder",23,5,3,0,"torch_holder_64")
torch_holder_64.controller:setHasTorch(true)
spawn("torch_holder",21,7,3,0,"torch_holder_65")
torch_holder_65.controller:setHasTorch(true)
spawn("torch_holder",23,7,1,0,"torch_holder_66")
torch_holder_66.controller:setHasTorch(true)
spawn("torch_holder",20,10,2,0,"torch_holder_67")
torch_holder_67.controller:setHasTorch(true)
spawn("torch_holder",21,9,3,0,"torch_holder_68")
torch_holder_68.controller:setHasTorch(true)
spawn("torch_holder",18,8,1,0,"torch_holder_69")
torch_holder_69.controller:setHasTorch(true)
spawn("torch_holder",18,7,3,0,"torch_holder_70")
torch_holder_70.controller:setHasTorch(true)
spawn("torch_holder",14,8,2,0,"torch_holder_71")
torch_holder_71.controller:setHasTorch(true)
spawn("torch_holder",15,7,3,0,"torch_holder_72")
torch_holder_72.controller:setHasTorch(true)
spawn("torch_holder",7,7,2,0,"torch_holder_42")
torch_holder_42.controller:setHasTorch(true)
spawn("torch_holder",5,7,3,0,"torch_holder_73")
torch_holder_73.controller:setHasTorch(true)
spawn("torch_holder",5,21,3,0,"torch_holder_74")
torch_holder_74.controller:setHasTorch(true)
spawn("torch_holder",5,21,1,0,"torch_holder_75")
torch_holder_75.controller:setHasTorch(true)
Example of a Challenge in JSON

This appendix shows an example of a challenge written in the json file.

Listing B.1: Example of a challenge

```json
{
  "challenges": [
    {
      "name": "Achiever1",
      "pattern": "PressurePlate[dungeon_pressure_plate = false false true false true [0 1 onActivate activate]];Spawner[spawner bottom rogue_boots];PressurePlate[dungeon_pressure_plate = false false true false true [2 1 onActivate activate]];Spawner[spawner bottom rogue_gloves]/Spawner[spawner top rogue_hood];PressurePlate[dungeon_pressure_plate = false false true false true [1 0 onActivate activate]];Spawner[spawner top rogue_pants];PressurePlate[dungeon_pressure_plate = false false true false true [3 0 onActivate activate]]/PressurePlate[
```
activate]; Spawner[spawner left rogue_vest]; ScriptEntity[
script_entity - [setComplete = 0 function checkSet() if (party.party:
getChampion(1):isArmorSetEquipped('rogue') or party.party:getChampion
(2):isArmorSetEquipped('rogue') or party.party:getChampion(3):
isArmorSetEquipped('rogue') or party.party:getChampion(4):
isArmorSetEquipped('rogue')) and setComplete == 0 then playSound('discover_spell') hudPrint('You have a complete Rogue set!')
setComplete = 1 end end]]; Timer[timer - - - true - [2 2 checkSet]]/O;
Chest[chest top - [Figure[figure_crowern -] Figure[
figure_ice_guardian -] Figure[figure_ogre -] Figure[figure_skeleton
-] Figure[figure_snail -] Scroll[scroll - [You have a set as a reward
in this room. Use the figures on the pressure plates to reveal each
piece. After all pieces revealed don't forget the figures. You don't
want to leave them behind!]]]; Door[dungeon_door_iron bottom - true];
X",
5  "seeker": "0",
6  "survivor": "0",
7  "daredevil": "0",
8  "mastermind": "0",
9  "conqueror": "0",
10  "socialiser": "0",
11  "achiever": "1"
12 }]
13 }
Information about Archetypes’ Values Calculations

This appendix shows all the information about the BrainHex archetypes’ values calculations.

Listing C.1: Information for archetypes’ values calculations

QUIZ - Page 2

01. "Exploring to see what you can find."
   [Seeker]
02. "Frantically escaping from a terrifying foe."
   [Survivor]
03. "Working out how to crack a challenging puzzle."
   [Mastermind]
04. "The struggle to defeat a difficult boss."
   [Conqueror]
05. "Playing in a group, online or in the same room."
   [Socialiser]
06. "Responding quickly to an exciting situation."
   [Daredevil]
07. "Picking up every single collectible in an area."
   [Achiever]
08. "Looking around just to enjoy the scenery."
   [Seeker]
09. "Being in control at high speed."
   [Daredevil]
10. "Devising a promising strategy when deciding what to try next."
    [Mastermind]
11. "Feeling relief when you escape to a safe area."
    [Survivor]
12. "Taking on a strong opponent when playing against a human player in a
    versus match."
    [Conqueror]
13. "Talking with other players, online or in the same room."
    [Socialiser]
14. "Finding what you need to complete a collection."
    [Achiever]
15. "Hanging from a high ledge."
    [Daredevil]
16. "Wondering what's behind a locked door."
    [Seeker]
17. "Feeling scared, terrified or disturbed."
    [Survivor]
18. "Working out what to do on your own."
    [Mastermind]
19. "Completing a punishing challenge after failing many times."
    [Conqueror]
20. "Co-operating with strangers."
    [Socialiser]
21. "Getting 100 (completing everything in a game)."
    [Achiever]

I love it! +2 pts
I like it. +1 pts
It's okay. +0 pts
I dislike it. -2 pts
I hate it! -4 pts

RATE - Page 3
01. "A moment of jaw-dropping wonder or beauty."
   [Seeker]
02. "An experience of primeval terror that blows your mind."
   [Survivor]
03. "A moment of breathtaking speed or vertigo."
   [Daredevil]
04. "The moment when the solution to a difficult puzzle clicks in your mind."
   [Mastermind]
05. "A moment of hard-fought victory."
   [Conqueror]
06. "A moment when you feel an intense sense of unity with another player."
   [Socialiser]
07. "A moment of completeness that you have strived for."
   [Achiever]

1 (worst) +2 pts
2 +4 pts
3 +6 pts
4 +8 pts
5 +10 pts
6 +12 pts
7 (best) +14 pts
This appendix presents part of the questionnaire used on the challenge validation test.
Challenge Validation

With this questionnaire, we are testing some challenges for the game Legend of Grimrock 2 (LoG2). Could you help us find which challenges are more interesting?

Answering this will take, approximately, 30 minutes. In advance, I want to thank you for your time.

* Required

Tester Characterization

This section asks about you and your gaming habits.

1. Age: *

2. Sex: *

   * Mark only one oval.
   - Male
   - Female
   - Intersex
   - Would rather not to say

3. How often do you play video games? *

   * Mark only one oval.
   - 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.

4. What is your professional relation with video games? *

   * Mark only one oval.
   - None.
   - I am a student in a course related to video games.
   - I am researcher in a field related to video games.
   - My profession is related to video games.

Legend of Grimrock 2

Legend of Grimrock 2 (LoG2) is a grid-based dungeon crawler RPG where you play as a party of four members.
5. Are you familiar with the grid-based dungeon crawler RPG genre?
   (e.g. Dungeon Master, Eye of the Beholder, Legend of Grimrock, Vaporum)
   Mark only one oval.
   — I don’t play video games.
   — I play video games but not of this genre.
   — I am familiar with the genre and played at least one game of the genre.
   — This genre is one of my favorites, and I played several games of this genre.

6. Have you played a game of the “Legend of Grimrock” series before?
   Mark only one oval.
   — No.
   — Yes.

Challenges
Please watch each video and state your agreement with the sentence written underneath.
There are links for a fullscreen version of the videos in each challenge, however, you can still watch
them directly here.

https://youtu.be/2BhNpqitaYA
12. "I like what this challenge encourages the player to do." *
Mark only one oval.

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strongly disagree</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Challenges - Sentences**

Please assign each sentence to one of the challenges presented below. The videos are the same videos you watched in the previous section of the questionnaire, but now have a number assigned. They are provided below the grid so you can rewatch them at your leisure.
13. Assign each sentence to a challenge. *
Mark only one oval per row.

<table>
<thead>
<tr>
<th>Challenge</th>
<th>#1</th>
<th>#2</th>
<th>#3</th>
<th>#4</th>
<th>#5</th>
<th>#6</th>
</tr>
</thead>
<tbody>
<tr>
<td>You like negotiating dizzying platforms or rushing around at high speed while you are still in control.</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>You like defeating impossibly difficult foes, struggling until you eventually achieve victory.</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>You like solving puzzles and devising strategies.</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>You like finding strange and wonderful things, or finding familiar things.</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>You like escaping from hideous and scary threats, pulse-pounding risks.</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>You like collecting anything you can collect, and doing everything you possibly can.</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
</tbody>
</table>

Challenge #1 ([https://youtu.be/2BhNpqitaYA](https://youtu.be/2BhNpqitaYA))
Please click on the following link ([http://survey.ihobo.com/BrainHex/](http://survey.ihobo.com/BrainHex/)) and answer to the BrainHex questionnaire. Then, come back to our questionnaire and tell us what results you did got, stating a value from -10 to 20 in each one of the following dimensions.

14. Achiever: *

15. Conqueror: *

16. Daredevil: *

17. Mastermind: *

18. Seeker: *
19. Socialiser: *

20. Survivor: *

Submission
Thank you for your participation in this test!
Participant’s Opinion Questionnaire

This appendix presents the questionnaire used on our final test.
Participant's Opinion #1

With this questionnaire, we are testing a level's content for the game Legend of Grimrock 2 (LoG2), taking into account what a player values most in a game. Could you give us your opinion about the level?

Answering this will take, approximately, 20 minutes. In advance, We want to thank you for your time.

* Required

Tester Characterization

This section asks about you and your gaming habits.

1. ID: *

2. Age: *

3. Sex: *

   Mark only one oval.
   - Male
   - Female
   - Intersex
   - Would rather not to say

4. How often do you play video games? *

   Mark only one oval.
   - 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.

5. What is your professional relation with video games? *

   Mark only one oval.
   - None.
   - I am a student in a course related to video games.
   - I am researcher in a field related to video games.
   - My profession is related to video games.

Legend of Grimrock 2

Legend of Grimrock 2 (LoG2) is a grid-based dungeon crawler RPG where you play as a party of four members.
6. Are you familiar with the grid-based dungeon crawler RPG genre? *
   (e.g. Dungeon Master, Eye of the Beholder, Legend of Grimrock, Vaporum)
   Mark only one oval.
   
   ☐ I don’t play video games.
   ☐ I play video games but not of this genre.
   ☐ I am familiar with the genre and played at least one game of the genre.
   ☐ This genre is one of my favorites, and I played several games of this genre.

7. Have you played a game of the “Legend of Grimrock” series before? *
   Mark only one oval.
   
   ☐ No.
   ☐ Yes.

Game Experience Questionnaire
This section asks you to indicate how you felt while playing the game for each of the items.
8. Core

Mark only one oval per row.

<table>
<thead>
<tr>
<th></th>
<th>not at all (0)</th>
<th>slightly (1)</th>
<th>moderately (2)</th>
<th>fairly (3)</th>
<th>extremely (4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>I felt content</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I felt skillful</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I was interested in the game’s story</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I thought it was fun</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I was fully occupied with the game</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I felt happy</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>It gave me a bad mood</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I thought about other things</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I found it tiresome</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I felt competent</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I thought it was hard</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>It was aesthetically pleasing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I forgot everything around me</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I felt good</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I was good at it</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I felt bored</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I felt successful</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I felt imaginative</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I felt that I could explore things</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I enjoyed it</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I was fast at reaching the game’s targets</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I felt annoyed</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I felt pressured</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I felt irritable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I lost track of time</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I felt challenged</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I found it impressive</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I was deeply concentrated in the game</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I felt frustrated</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>It felt like a rich experience</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I lost connection with the outside world</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I felt time pressure</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I had to put a lot of effort into it</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Submission
Thank you for your participation in this test!