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.

I. INTRODUCTION

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. Rogue¹, Minecraft², Galaxy Arms Race [3] and Diablo³ are some examples of these applications. However, 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.

So, with the use of procedural content generation, player’s models and modified story arcs, how do we make content placement tailored to the player’s individual characteristics?

This is where our idea comes in. We will place the content by crossing it with an interest curve⁴, 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 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 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. This model aims 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 evaluate our model’s content placement, we conducted two different tests: a first one to validate the challenges used and a second one to obtain the participants’ opinions about the level’s content. With these tests’ results we were able to draw conclusions that support our hypothesis.

II. RELATED WORK

A. Personality Models

A person’s personality is directly related to the way he/she acts upon the world. Each person has different behavior,

¹Instituto Superior Técnico, Campus Taguspark, Av. Prof. Dr. Cavaco Silva, 2744-016 Porto Salvo, Lisboa, Portugal 
³Mojang, “Minecraft”, 2009
⁴Blizzard North, “Diablo”, Blizzard Entertainment, 1997
⁵Blizzard North, “Diablo”, Blizzard Entertainment, 1997
⁶Modified story arc with distance as the XX axis and interest as the YY axis
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. To better understand a person’s personality, personality models were created.

Myers-Briggs Type Indicator [4] 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. This model categorizes personality in 16 types: 4 main psychological functions of consciousness by which people experience the world - sensation, intuition, thinking and feeling - that can vary according to 2 attitude types: extraversion and introversion. Judging and perceiving were added to explain the interaction with the outside world.

Five Factor Model [5] 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.

Cloninger’s Temperament and Character Inventory [6] appears as a 240-item questionnaire designed to explain the unique personality of an individual, identifying the intensity of and relationships between 7 personality dimensions, 4 of which being Temperament - Novelty Seeking, Harm Avoidance, Reward Dependence and Persistence - and the other 3 being Character - Self-Directedness, Cooperativeness and Self-Transcendence.

B. 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. To identify the types of players, player’s models were created.

To create the Bartle Player Types [7] a study was conducted to classify players according to their preferred in-game actions. The study analysis showed four different pattern types that lead to 4 different characters: Achievers, Explorers, Socialisers and Killers.

Demographic Game Design [8], based on applying Myers-Briggs typology to data gathered about the players and their gameplay needs. This model focus on market oriented game design. It has some similarity with Bartle Player Types, identifying 4 player types: Conquerors, Managers, Wanderers and Participants.

BrainHex is a player satisfaction model that was created by Nacke et al. [9] and is based on the results from the earlier Demographic Game Design and neurobiological studies. This model divide players into 7 different archetypes: Seeker, Survivor, Daredevil, Mastermind, Conqueror, Socialiser and Achiever. There are also 7 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.

Nick Yee and Nicolas Ducheneaut, Co-Founders of Quantic Foundry, came up with an empirical Gamer Motivation Model [10] through the gathering of data from over three hundred thousand gamers and using a psychology research standardized process that revolves around factor analysis. This model identifies twelve motivation factors: Destruction, Excitement, Competition, Community, Challenge, Strategy, Completion, Power, Fantasy, Story, Design and Discovery.

C. Procedural Content Generation

Togelius et al. [11] describe Procedural Content Generation (PCG) as the algorithmic creation of game content with limited or indirect user input. According to Togelius et al. [11], 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. There are numerous ways to use PCG. This being said, we focused our work on a specific part of PCG, content placement. We researched for works related to PCG and selected Experience-Driven Procedural Content Generation and 3Buddy.

Yannakakis et al. define Experience-Driven PCG (EDPCG) [12], 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. 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). Content quality evaluation is done, using the acquired player models, in the content generation phase. The Content Quality component is divided in three key evaluation functions: direct, simulation-based and interactive. Game content representation is a central question in EDPCG and can be symbolic or subsymbolic. 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 experience, ideally, identifying if content should be generated for that player.

Lucas and Martinho created 3Buddy [13], 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.

D. Progression

Pereira and Martinho created a progression model [14] that has the goal to evaluate the player’s skill level. This model uses a game’s mechanics, challenges and pace 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 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 [15], 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 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. Aside from the characters and the player, 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. 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, and then selected based on which has the highest priority to the required part of the sequence.

E. 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 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 [9], 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 [13] 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 [12] 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 [15] 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 [15] beat scoring method, we will use an interest curve\(^6\) to place content relevant to the player.

III. COMPUTATIONAL MODEL

A. The Four Components

1) Challenges and their Grammar: 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 be later interpreted and used by our model. Each challenge has 9 attributes: name, pattern, seeker, survivor, daredevil, mastermind, conqueror, socialiser and achiever. Name is a string with the challenge’s name. Pattern is a group of individual content from Legend of Grimrock 2, where a grammar is needed. The remaining attributes are the BrainHex’s archetypes. Next, we present an example 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]

With these grammar we created 6 challenges, one for each of the BrainHex archetypes, except socialiser, since the game chosen for this is single player, so there are no interactions with other players.

\(^6\)instead of tension, we cross the story arc with player’s interest or personality compatibility
2) Player Model and Archetypes Calculations: We used the BrainHex player model, that 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.

3) Interest Curve: As previously mentioned, we adapted the idea from Façade [15] 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. 1 we have the interest curve used during the final evaluation of our thesis.

![Interest curve](image)

Fig. 1. Interest curve

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. 2 shows both levels’ layouts side by side.

![Original and modified level layout](image)

(a) Original level layout  (b) Our level layout

Fig. 2. Original and modified side by side

B. Model’s Architecture

Our content placement algorithm uses the four aforementioned components throughout a run, as seen in Fig. 3. Before our algorithm’s run starts, we import a level that is analyzed and parsed. This level is written in a text file 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:

\[
 W_0 = \frac{P_0}{\sum P_i} \tag{1}
\]

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

\[
 U_k = \sum W_i O_i \tag{2}
\]

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

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:

\[ V_k = (U_k - d_k)^2 \]  

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 (\( V_{\min} \)).

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.

IV. MODEL IMPLEMENTATION

A. 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.

B. 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.

C. Player’s Interest

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 1). Afterwards, for each challenge, our algorithm multiplies each archetype’s weight by the respective 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 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.

D. 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. 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” 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, 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. This is repeated until there are no more pattern’s rotations available.

\[ 7 \text{if no center ("#") is read from the pattern, we assume the center is the first position } (i, j) = (0, 0) \]

\[ 8 \text{all comparisons return true} \]
to be done once, being only necessary to sort the interests in the cells corresponding to the points of the interest curve.

E. Content Sorting

Having filtered which challenges can be placed in what cells, we finally use the interest curve’s values. 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). 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.

F. 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.

G. 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.

1) 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.

2) 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, 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.

H. 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, allowing testers to enjoy our algorithm’s content placement.

V. EVALUATION

A. 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 and asked players to help us identify each one. The duration of this validation was between 20 and 30 minutes. To acquire all the needed information to validate our challenges we used a questionnaire and asked 16 participants to answer it.

1) 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 I shows the result of our data processing.

<table>
<thead>
<tr>
<th>BrainHex</th>
<th>Achiever</th>
<th>Conqueror</th>
<th>Explorer</th>
<th>Fighter</th>
<th>Healer</th>
<th>Seeker</th>
<th>Survivor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Achiever</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>Conqueror</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>Explorer</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>Fighter</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>Healer</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>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>Survivor</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>

In table I, 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:

\[
result = video - (((BrainHex + 10) / 7.5) + 1)
\]  

where \(result\) is a value in the range \([-4, 4]\), \(video\) 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 \(((\text{BrainHex} + 10)/7.5) + 1\) 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 I, 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 I, 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.

2) **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 II 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</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>02</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>03</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>04</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

**Table II**

SECOND ITERATION WITH 4 PARTICIPANTS

In table II, 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.
- 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.

B. **Final Evaluation**

Having the challenges validated and implementation completed, we were ready for the final tests, where we would

---

9 each pair personal, general counts as 1 answer
try to verify our hypothesis. To test our model, we asked 30 participants 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. 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. Since we had the core module of the game experience questionnaire [16] as part of our test, we used their scoring guidelines. The core module of the game experience questionnaire [16] assesses game experience as scores on seven components: competence, sensory and imaginative immersion, flow, tension/annoyance, challenge, negative affect and positive affect. The results obtained in each component are presented from Fig. 4 to Fig. 10. Table III shows a summary of the results obtained in each component through the questionnaire - Range, Min, Max, Mean and Standard Deviation - as well as the results from a Mann-Whitney test - Mean Rank, Z and p values.

With the data shown in the figures - Fig. 4 to Fig. 10 - and Table III, we can withdraw several conclusions. We fail to reject the null hypothesis in Competence, since $Z = -0.417$ and $p = 0.676$. Given that the challenges are the same but presented in a different order, the overall competence is expected to be no different. The Immersion component fails to reject the null hypothesis, having $Z = -0.815$ and $p = 0.415$. Given that the challenges are the same but presented in a different order, the overall Sensory and Imaginative Immersion is expected to be no different.

<table>
<thead>
<tr>
<th></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>1.97</td>
<td>1.01</td>
<td>14.83</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>14.20</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>2.80</td>
<td>0.80</td>
<td>3.80</td>
<td>2.33</td>
<td>0.78</td>
<td>11.73</td>
<td>-2.363</td>
<td>0.018</td>
</tr>
</tbody>
</table>

TABLE III
SUMMARY OF THE FINAL EVALUATION RESULTS

10There is no difference between the ranks of the two Competences.
we also fail to reject the null hypothesis, since $Z = -0.500$ and $p = 0.617$, we also fail to reject the null hypothesis in Flow. With the shown data, we do not have enough evidence to conclude that the difference between both Tensions is statistically significantly ($Z = -1.604$ and $p = 0.109$). So, we fail to reject the null hypothesis. However, there seems to be a tendency to the increase of one Tension, which may become statistically significant with more participants. In Challenge, we also fail to reject the null hypothesis, since $Z = -0.500$ 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. By looking at the data shown in the figures and table, we do not have enough evidence to conclude that the difference between both Negative affects is statistically significantly ($Z = -0.126$ and $p = 0.900$). So, we fail to reject the null hypothesis. With this data, $Z = -2.363$ and $p = 0.018$, we reject the null hypothesis and have enough evidence to conclude that the difference between both Positive affects is statistically significantly.

We were also able to obtain anecdotal information, through the short interview, 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.”

C. 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 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 [16], 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.

VI. CONCLUSIONS

With this work 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 [4], Five Factor Model [5] and Cloninger’s Temperament and Character Inventory [6]. In players models, we presented Battlete Player Types [7], Demographic Game Design [8], BrainHex [9] and Quantic Foundry’s Gamer Motivation Model [10]. In procedural content generation, we described Experience-Driven Procedural Content Generation [12] and 3Buddy [13]. Lastly, in progression, we presented Façade [15]. 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 [16].

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.

A. 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.

ACKNOWLEDGMENT

The author would like to thank the supervisor of this work, Prof. Carlos Martinho for the guidance and availability. A special thanks to Diogo Rato, who supported me during this work. Thank you to all the participants, who helped in the challenges’ validation and this work’s final evaluation and to the people that somewhat contributed to this work.

REFERENCES


