Modelling Player Skill in a Procedurally Generated Single-Player Videogame

Jorge Nunes
jorge.costa.nunes@tecnico.ulisboa.pt
Instituto Superior Técnico
Portugal

ABSTRACT

Difficulty is one of the biggest motivational pulls of single-player video games, hence its importance on how to evaluate player skill and difficulty scaling, while improving player experience and keeping the state of flow. In this article, we explore how an Elo system can be used in order to create a Dynamic Difficulty Adjusted system in a bullet-hell single player Procedural Generated video game called Holiday Knight. Our approach will test if the player feels more enjoyment when playing a PCG video game while its difficulty is being adapted according to their skill as they play, versus a version of the same game where difficulty is linearly increasing and a version where challenges are randomly selected. In our evaluation, participants (N = 30) played the game and answered a questionnaire that focused on measuring the player's intrinsic motivation, perceived affectivity, as well as which version they preferred playing. We were not able to find statistically significant differences that could support which was the preferred version, but the players were able to identify the randomness of the version where challenges are randomly selected and the determinism of the versions where the difficulty of the challenges is adapted to the player's performance and the version where difficulty is linearly increasing.

CCS CONCEPTS

• **Computing methodologies**;

KEYWORDS

flow, single-player, procedural content generation, dynamic difficulty adjustment; player skill; subjective experience measurement

1 MOTIVATION

The gaming industry revenue is increasing year by year with an all-time high digital games spending of $127 billion across mobile, PC and console platforms in 2020[1] and a market forecast to be worth $256.97 billion by 2025[2].

Being such a rising market, video game development techniques are also evolving: Procedural Content Generation (PCG) allows the designer to generate more content and increase variety in the game being developed, which leads to having more replayability and therefore more engagement from the player [Smith 2014].

Procedurally generated content takes form in multiple ways, such as No Man's Sky[3] which is able to generate over 18 quintillion planets; Borderlands 3[4] gives the player 1 billion unique weapons to choose from; The Binding of Isaac: Rebirth[5] is composed of floors which are arranged from a huge amount of rooms put together respecting certain rules; Ape Out[6], just like the previous mentioned game, procedurally generates its floors, but also its music, which changes according to the player's actions in game.

Besides content variety, another key aspect video games must take into account to keep the player engaged is difficulty. This can be static, like in The Last of Us[7], where difficulty is set upon creating a new save file and the player is able to select one from a total of 5 different difficulty levels, which affect parameters such as enemies damage, supply spawn rates and player abilities, etc. Difficulty can also be adapted to the player's skill, like in Mario Kart[8], where its Rubber Band AI will cause computer-controlled racers to match the player's pace, making it so they are always at least threatening to get back in the race[9].

2 PROBLEM

Can we convey a fairer experience when dynamically adapting the difficulty of a PCG single player video game?

Similarly to the multiplayer context, where the player hopes their adversaries to have a closely matching skill level, a fair experience in a single player setting is conveyed when the challenges presented are appropriate to the player's skill.

There are no two players that are the same: they have different past gaming experiences, skill and learning rates. Furthermore, players might be looking for different gaming experiences: one might be looking for a more "chill" playthrough, while another might be going for a more "hardcore" one.

It's not an easy task to set the difficulty level on a designed game and even harder one to do it on a single player video game in which every run is different from the other, since it is being procedurally generated.

[1] https://www.gamesindustry.biz/articles/2021%2D01%2D06%2Ddigital%2Dgames%2Dspending%2Dreached%2DUSD127%2Dbillion%2Din%2D2020
[2] https://techjury.net/blog/gaming-industry-worth/#gref
[3] No Man's Sky (PlayStation 4, PC, 2016; Xbox One, 2018), a survival action-adventure video game, developed and published by Hello Games.
[4] Borderlands 3 (PlayStation 4, PC, Xbox One, Google Stadia, 2019), an open world role-playing first-person shooter video game, developed by Gearbox and published by 2K Games.
[5] The Binding of Isaac: Rebirth (PC, PlayStation 4, PlayStation Vita, 2014; Nintendo 3DS, Wii U, Xbox One, 2015; iOS, Nintendo Switch, 2017; PlayStation 5, Xbox Series X/S, 2021), an indie rogue-like dungeon crawler video game, developed and published by Nicalis.
[6] Ape Out (Nintendo Switch, PC, 2019), a single player beat 'em up video game, developed by Gabe Cuzzillo and published by Devolver Digital.
[7] The Last of Us (PlayStation 3, 2013; PlayStation 4, 2014), an apocalyptic action-adventure video game, developed by Naught Dog and published by Sony Computer Entertainment.
[8] Mario Kart (Nintendo platforms), a kart-racing video game series, developed and published by Nintendo.
[9] https://www.svg.com/138490/games-you-didnt-know-featured-dynamic-difficulty/

3 HYPOTHESIS

Multiplayer video games often offer Player Versus Player (PVP) scenarios where groups of players are put against each other. Matchmaking systems tend to be fair and form teams of even skill level, but to avoid longer queue times, the skill gap is widen. This leads to scenarios where one group can clearly outmatch the other, affecting negatively both teams: the winner will feel the match to be too easy and become bored, while the one who lost will feel the match to be unfair and become frustrated. In order to tackle this issue, a matchmaking system called Elo is used.

The Elo system is widely and successfully used in a variety of games such as Rocket League[10], League of Legends[11] and Counter Strike: Global Offensive[12]. The one thing all the mentioned games have in common that allow for the Elo system to work is their multiplayer basis, which matches up players versus players, each with their own skill level and, after match completion, results in a skill level update.

To answer the stated problem, we will be modifying Holiday Knight [Pardal 2019], a PCG single-player video game in which the player goes through a number of rooms while shooting their way through enemies (depicted in **fig. 1** and further described in **section 5**), and dynamically adapt its difficulty according to the player's skill using an Elo system, where challenges are seen as players. This method seeks to keep an experience challenging enough that the player finds it intriguing and feels fulfilled after completing it, but not too difficult that it becomes frustrating. The idea is to keep the player engaged and engrossed while also steadily improving their skill.

Figure 1: Holiday Knight's tutorial room

We hypothesize that:

- **H1**: The player will prefer a version of the PCG single player video game where challenges are ordered based on their difficulty (NA version), to a version where challenges are randomly selected (R version);
- **H2**: The player will prefer a version of the PCG single player video game where challenges are adapted to an Elo-based model measuring player's skill (A version), to the R version;
- **H3**: The player will prefer the A version, to the NA version.

In order to test these hypothesis, we will be developing three versions of Holiday Knight, in which the challenges the player will face will be the enemies in each room.

The three formulated hypothesis will test the A version to be the most preferred version, followed by the NA version and finally by the R version. This conclusion will be withdrawn by analyzing the feedback provided by the players upon trying all three versions and comparing the gaming experience from each.

4 RELATED WORK

4.1 Cognitive Flow

Cognitive Flow captures the positive mental state of being completely absorbed, focused, and involved in your activities at a certain point in time, as well as deriving enjoyment from being engaged in that activity[13].

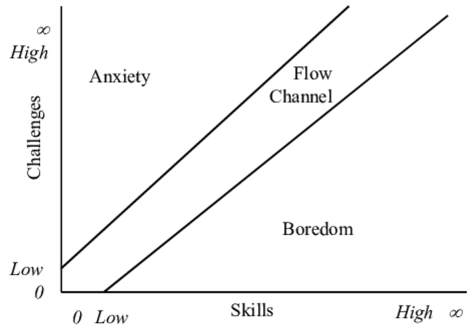

Figure 2: Flow Channel according to Csikszentmihalyi

In **fig. 2**, we see the player enters a zone of anxiety when facing a challenge with a level that surpasses their skill, which would make them frustrated and end on giving up on the game. Opposing to this, when the player's skill is greater than the level of the challenge, we would get what would turn up to be an easy challenge and the player would get bored, eventually getting tired of the game and, once again, giving up on it.

4.2 Procedural Content Generation

Procedural content generation is the programmatic generation of game content using a random or pseudo-random process that results in an unpredictable range of possible game play spaces[14]. In other words, PCG in games is the creation of game content with limited or indirect input from the designer. This allows for a possible infinite number of generated content in a game, instead of the

[10]Rocket League (PC, PlayStation 4, Xbox One, 2015; Nintendo Switch, 2017), a vehicular soccer video game, developed and published by Psyonix.
[11]League of Legends (PC, 2009), a multiplayer online battle arena video game developed and published by Riot Games.
[12]Counter Strike: Global Offensive (PC, PlayStation 3, Xbox 360, 2012), a multiplayer first-person shooter video game, developed by Valve and Hidden Path Entertainment and published by Valve.

[13]https://positivepsychology.com/what-is-flow/
[14]http://pcg.wikidot.com/what-pcg-is

only ones that were prepared by the designers themselves, thus increasing game replayability.

Togelius et al. [Shaker et al. 2016] state PCG as a more creative way for designers to produce game content, while also reducing the man-power and time needed to do so. Designers are now able to create huge open-worlds upon defining a set of rules and thus obtaining multiple suitable scenarios. This even allows the retrieving of a solution the designers didn't think themselves. Such scenario is looked into in Brown and Maire's work [Browne and Maire 2010] regarding evolutionary game design, where they study both a system's ability to automatically measure generated games and their potential to interest human players, and its ability to create new high-quality games.

4.2.1 The Properties of PCG .

We can think of implementations of PCG methods as solutions to content generation problems. The desirable or required properties of a solution are different for each application. The only constant is that there are usually tradeoffs involved. In his work, Togelius et al. [Shaker et al. 2016] states a list of common desirable properties of PCG solutions:

- Speed;
- Reliability;
- Controllability;
- Expressivity/Diversity;
- Creativity/Believability.

4.3 Difficulty in Video Games

One of the fundamental issues to tackle in the design of video games is mostly referred to as creating a well-shaped difficulty curve. This means that one of the core element of a good game design is to make the game just as difficult as it has to be, so that the player feels challenged enough, but not too much.

4.3.1 Static Difficulty .

In video games where difficulty is static, players face different challenges with a set difficulty in order to improve their skill. When the player is skilled enough to surpass a certain challenge, they get more skilled and are able to face harder challenges. In this case, the player must adapt and face the video game's difficulty in order to progress.

In their work, Aponte et al. [Aponte et al. 2011] study the choice of challenges and how their succession is related to the flow principle (see section **4.1**). At any time of the game, the difficulty of the next challenge must be a little higher than the current level of the player apprenticeship. When they win the challenge and the tension decreases, the player gets new skills and abilities. This correlated progression of skills and difficulty must be kept all along the game.

4.3.2 Dynamic Difficulty Adjustment .

In a video game, rather than presenting every player with the same difficulty, some games tailor the difficulty presented to each player using dynamic difficulty adjustment (DDA). DDA is a general term for techniques used to dynamically modify in-game difficulty during the course of gameplay as a means to tailor the experience more toward the player's current level of play, without them noticing the adaptation. This has been achieved in games through various techniques such as parameter tuning, modifying level design, machine learning, the use of rating systems and player modeling.

In her work, Hunicke [Hunicke 2005] studied how even crude adjustment algorithms can improve performance, while retaining the player's sense of agency and accomplishment. Furthermore, confirmed that the adjustment of even the most basic gameplay elements is nearly imperceptible to the player.

4.4 Player Skill and Progression Models

Player skill is defined as the amount of competence a player has over a certain activity or collection of tasks, and player progression is the measure of how those skills improve over time. In video games, a player's abilities are impacted by their experiences, circumstances, and background. Players are different and may have different goals towards specific video games: a hardcore player will try their best to perfect each technique and mechanism the game presents, while a casual player might just want to spend some moments of fun and relax with no intention of stressing over not being perfect. Despite their goal, players also progress at different rates: one might have a natural talent towards a certain game genre or the challenge might require a skill which the player is already good at (having a fast reaction time, spotting something on the environment or figuring out a clue) that another does not have.

As the player masters a video game's mechanics, their skill will increase, thus being able to face tougher challenges and having easier ones pulling them away from the flow state. In order to adapt a video game's difficulty, we then need to be able to measure player skill and how it is progressing.

4.4.1 Player Model .

In his work, Cook[15] defines the player as an entity that is driven, consciously or subconsciously, to learn new skills high in perceived value. They gain pleasure from successfully acquiring skills.

There's a total of three key concepts in this player model:

- **Skill** - The behavior the player used to interact and manipulate the world;
- **Drive to Learn** - Playing is part of the human instinct: when in a safe environment, people will begin playing by default;
- **Perceived Value** - Players pursue skills with high perceived value over skills with low perceived value.

4.4.2 Bicho's Skill-Based Progression Model .

"Go, Go Hexahedron" is an endless running side-scrolling video game developed by Bicho in his thesis [Bicho 2018] and article[Bicho and Martinho 2018], which can be seen in **fig. 3**. Here, the player is controlling a white cube that is continuously "running" towards the right of the screen, with the objective of obtaining the most points, while avoiding every obstacle on the way, in one of the two available paths.

[15] https://www.gamedeveloper.com/design/the-chemistry-of-game-design

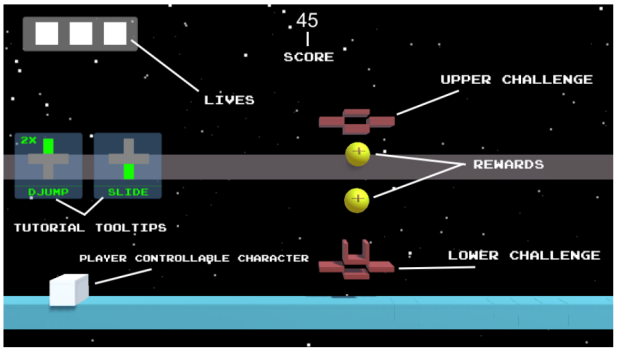

Figure 3: *Go, Go Hexahedron* components

The player has a total of four mechanics to use in game: the single jump, double jump, dash and slide. Other than these, the player can also use the switch mechanic in order to switch the obstacle provided in both paths from one to the other.

Difficulty increases by individually changing the size of each of the six available challenges. This model, however, is not limited to size, with the only requirement being to have a minimum and maximum value that can represent a challenge, like how fast a slicing blade is or the number of objects in a challenge. The value used for interpolation between the two extremes of a certain dimension of a challenge is calculated using a logarithmic function, which is based on the number of times the challenge was spawned in the current game. This ensures a logarithmic progression of difficulty, where each update is smaller as it gets closer to what would be humanly possible through the character controls.

To compute skill, each attempt at a challenge was recorded, expressed as a success rating, ranging from 0 to 1, measuring how the player performed. The list associated with each pair $< challenge, mechanics >$ represents a sliding window recording the success ratings of the most recent attempts. Every time the player attempts to overcome a challenge using a specific combination of mechanics, the new success rating is inserted in the respective sliding "window". The player's skill is measured in each pair is calculated through a weighted arithmetic mean.

4.4.3 *Elo Statistical Model for Gaming Data .*

Elo related ranking systems are popular in multiplayer games, which often place the players in a queue while the system uses a variety of measures such as ping, time already spent waiting and Elo rating to match together. Despite its popularity among video games, some researchers have developed Elo based models in predicting animal behavior [Newton-Fisher 2017], detecting deficiencies in fabric patterns [Tsang et al. 2016] and student performance evaluations [Brinkhuis and Maris 2009] [Pelánek 2016].

The Elo formula is a simple and malleable way to rank skill. Its main premise is that a win will increase one's skill rating, while a loss will decrease it. How much a player's skill rating changes is a function of their calculated probability of winning the match, which is based on their opponent's skill rating. If their probability of winning is around 50%, the change in the player's skill rating is relatively equal in either the positive (winning) or negative (losing) direction. However, if a player with a much lower skill rating is competing against a much higher rated player, the lower skilled player will gain a higher amount to their skill rating for a win than the higher skilled player to account for the lower skilled player's lower probability of winning.

The probability of a player winning is a logistic function, with the typical feature of the probabilities of a win less than 100% even when a very high skill player is paired with a very low skilled one. Subsequently, the chances of a win can never be 0%. If P represents the player's current skill rating and O is the opponent's skill rating, then one calculates E, the expected probability of winning, by:

$$E = \frac{1}{1 + 10^{-(P-O)/a}} \tag{1}$$

In **eq. 1**, a is usually 400. The value of a can change depending on the skill ranking of the player and helps with the issue that the winning probability for high rank players is occasionally overestimated by the formula. The value 10 is an additional parameter that represents the scaling of the skill rating that may be used.

The probability of an expected win is then used when calculating the change in skill rating to be applied to a player's current skill estimate. To calculate the new skill rating *New*, let P represent the current skill rating, let *Out* be the outcome of the match (1 for a win and 0 for a loss), and let E be the probability of winning the match and K is a constant, frequently 32 in chess. The formula for the new skill rating is:

$$New = P + K\,(Out - E) \tag{2}$$

Eq. 2 shows that the more unexpected the outcome of a match (the greater the difference between *Out* and E), the larger the adjustment to the player's rating.

4.5 Subjective Experience Measurements

4.5.1 *Intrinsic Motivation Inventory.*

The Intrinsic Motivation Inventory (IMI)[16] is a multidimensional measure device intended to assess the participants on a subjective experience. It has been used in several experiments that are related to intrinsic motivation and self-regulation. This instrument assesses participants over six dimensions:

- Interest/Enjoyment;
- Perceived Competence;
- Effort/Importance;
- Pressure/Tension;
- Perceived Choice;
- Value/Usefulness.

The IMI consists of varied numbers of items from these subscales, all of which have been shown to be a factor analytically coherent and stable across a variety of tasks, conditions, and settings.

The order effects of item presentation appear to be negligible, and the inclusion or exclusion of specific subscales appears to have no impact on the others. Thus, it is rare that all items have been used in a particular experiment. Instead, experimenters choose the subscales that are relevant to the issues they are exploring.

The IMI items are often modified slightly to fit specific activities.

[16] https://selfdeterminationtheory.org/wp-content/uploads/2022/02/IMI_Complete.pdf

4.5.2 *Positive and Negative Affect Schedule.*

The Positive and Negative Affect Schedule (PANAS)[17] is a self-report questionnaire that consists of two 10-item scales to measure both positive and negative affect.

Positive and negative affectivity are human characteristics that describe how much people experience positive/negative affects, such as sensations, emotions and sentiments, and as a consequence, how they interact with others and their surroundings.

Researchers extracted 60 terms shown to be relatively accurate markers of either positive or negative affect, but not both. In other words, they chose terms that met a strong correlation to one corresponding dimension but exhibited a weak correlation to the other. Through multiple rounds of elimination and preliminary analyses with a test population, the researchers arrived at 10 terms for each of the two scales, as follows:

- **Positive affect:** Attentive, Active, Alert, Excited, Enthusiastic, Determined, Inspired, Proud, Interested, Strong;
- **Negative affect:** Hostile, Irritable, Ashamed, Guilty, Distressed, Upset, Scared, Afraid, Jittery, Nervous.

The scoring for the PANAS consists on comparing the obtained positive affect score and the negative affect score.

These are calculating by adding the score of the items respective to each scale. Higher scores represent higher levels of that affectivity, while lower scores represent lower levels.

5 TESTBED GAME

5.1 Holiday Knight

Holiday Knight is a Christmas themed bullet hell dungeon crawler, where the player faces a series of rooms, each containing a different enemy team. In order to progress, all enemies present in the current room must be eliminated, while dodging their shots.

From the moment the game starts, the player enters the first room and is presented what inputs will reflect on character actions such as moving, aiming, shooting, reloading and throwing their weapon.

For the next 20 rooms, the player must defeat all the enemies present in the current room, while trying to obtain the highest number of points. In order to do so, the player will have 7 health points, which are restored when advancing to the next room.

If the player manages to clear the room, they will earn a certain amount of points depending on the time they took to clear it. In case the players lose their 7 health points, the character dies, the enemies and all bullets are removed from the room and the character is then revived with its health back and no points earned for that specific room.

5.2 Game Environment

The game's environment is a major factor on the player's experience. The level design of the rooms and how the enemies are portrayed highly influence the player's immersion and flow state, as these were the components they would interact with the most.

5.2.1 *Rooms.*

In every run of the game, the player will always find themselves on a similar initial room, which is depicted in **fig. 1**.

[17] https://en.wikipedia.org/wiki/Positive_and_Negative_Affect_Schedule

There are a total of 4 different room environments: neutral, fire, grass and water. The player will be able to distinguish them by the color they have and some cosmetics, with the neutral room being brown, the fire room, being red, the grass room being green and the water room being blue.

The effects of each room environment are discussed on 5.2.3.

5.2.2 *Enemies.*

There are a total of 10 different enemies, which can be seen in **fig. 4**, named: Big Zombie, Chort, Ice Zombie, Imp, Masked Orc, Muddy, Orc, Shaman, Skelet and Swampy.

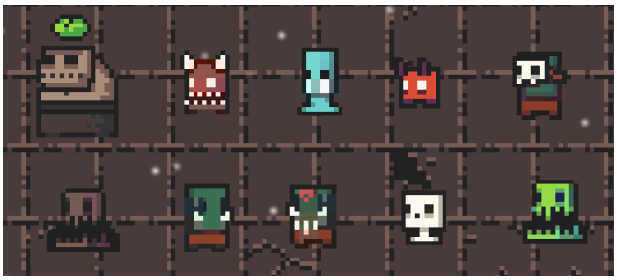

Figure 4: The 10 different enemies present in the game

All enemies have a similar behavior where they keep shooting at a specific fire rate, except the Ice Zombie, which throws itself against the player. Other than this shooting characteristic, all enemies move in order to get closer to the player until they get to a certain distance and stop. If the player tries to run away, they will keep following them to the closest point of the player within a certain distance.

Each enemy has multiple parameters, which vary from each other: stopping distance to the player, shooting pattern, fire rate, moving speed, health pool, damage per shot and type, which is defined by the color of its aura.

5.2.3 *Enemy Type and Room Environment Synergy.*

When an enemy of a certain type is generated in a room with a certain environment, its damage, health and speed parameters may be affected. These changes happen according to the type advantage: fire beats grass, grass beats water and water beats fire.

If an enemy type has advantage over the room environment, their stats are upgraded. If it's the room environment that has the advantage over the enemy type, then their stats are downgraded. In case the type and environment are the same, the enemy parameters are not affected.

6 IMPLEMENTATION

The developed versions of this work were implemented in Unity[18].

The programming language that's used in Unity and that was used in the project is C#, which is an object-oriented programming language.

In order to test the stated hypothesis, all three versions of Holiday Knight had to be implemented, making sure a wide variety of challenges is created for players with different levels of skill, while keeping the game fun and engaging.

[18] https://unity.com/

In Holiday Knight, a challenge is represented by an enemy team. This means in a single run, a total of 20 challenges are generated and must be selected for each room.

Despite this generation and selection not taking such a big part on the R version since these are random, on both A and NA versions they are a major role. Being able to classify and order the difficulty of each generated enemy team is core in the NA version and, similarly, being able to select an enemy team that matches accordingly to the player's performance in the A version.

Furthermore, to increase the number of challenges that can be generated, techniques such as dynamic challenge adaptations in the forms of enemy types and consequent variable parameters were implemented.

6.1 Dynamic Challenge Adaptation

An enemy team may be comprised of 2 to 4 enemies, which are chosen from the 10 different enemies shown of **fig. 4**. With the dynamic challenge adaptations, the number of different generated challenges is increased from 5850 to over 128 billion, thus avoiding the player from facing the same challenge twice throughout their whole experience.

Enemies's damage, health and speed parameters have five different levels: *Reduced*, *Low*, *Normal*, *High* and *Boosted*. When generated, each of the parameters can only be *Low*, *Normal* or *High*. For an enemy to have one (or more) of its parameter in the Reduced level, the generated level of the parameter has to be *Low* and the enemy's type must be a disadvantage over the environment type. Likewise, for an enemy to have a *Boosted* parameter, its generated level has to be *High* and the enemy's type have an advantage over the environment type.

In order for these changes on the enemies to be noticed by the player, visual differences were added. The higher the level of an enemy's damage parameter, the darker the bullet is. Regarding an enemy's health, the closer to the purple hue, the more health the enemy will have.

6.2 Challenge Generation

A *.csv* file was created where we could permanently store a number of challenges. We settled for a total of 1000 different enemy teams, since a number higher than this would need more time, memory and computational power. With a number less than 1000, there would be a chance of missing out on certain enemy teams and create gaps in these challenges' difficulty.

The generation process follows:

- Selecting number of generated challenges;
- Selecting number of enemies in each challenge;
- Selecting enemies for each enemy team;
- Selecting the base level for each of the enemies' parameters;
- Selecting the enemy type for each enemy.

6.3 Battlegrounds Scene

After generating an enemy team in each line of the *.csv* file, we need to assign a difficulty level to each of the challenges, so these can be selected for intended rooms for each of the following models.

In order to define if an enemy team is stronger than another, we developed a combat simulator with a set heuristic which had to be accurate in at least 80% of the cases.

To evaluate if the heuristic correctly predicts which of the enemy teams is the strongest, we developed a scene that works like a battleground. Here, enemies can be pre-selected for each of the teams with any level of parameters, type, room environment and even the spawn point in the room. This allows us to observe in real-time the fight between the two enemy teams.

In order to obtain the values for the heuristic calculation that will be referred next, the battlegrounds' scene was used so each of the ten available enemies was put against each other one thousand times for a total of ten thousand fights for each enemy. In each fight, the selected enemies were able to have their dynamic parameters in all five levels, so most of the possible combinations of each enemy would be spawned and registered if it would result in a win or loss. The information regarding percentage of wins for each enemy and the percentage of victories for each of the five levels of each dynamic parameter, was collected on two *.csv* different files.

6.4 Challenge Selection Heuristic

Simulating every needed fight between enemy teams would result in big waiting time intervals, hindering the player experience. To avoid this, we needed to find a better way to find the most difficult challenges and the correspondent strongest enemy teams. This is where the heuristic comes in, since it is a design-based formula that takes into account variables such as the number of enemies in a team, what specific enemies they are and with what parameters.

The heuristic calculates the skill of each enemy team. The one with the highest calculated value is deemed to be strongest and hence the hardest challenge.

After being able to calculate heuristic values, each enemy team is, using the formulated heuristic, put against the other 999 teams and accumulate as many wins as possible. With each enemy team having a certain number of wins, they can now be ordered according to their number of wins, representing its difficulty level.

6.5 Heuristic Validation

Although we were able to develop a heuristic that allows us to define the difficulty of each challenge, we must make sure it accurately represents the fights between each of the enemy teams and obtain the proposed 80% of accuracy, as mentioned in **section 6.3**.

In order to calculate the heuristic accuracy, a total of 50 fights between enemy teams was generated in the battlegrounds' scene and registered in a *.csv* file. Each of these fights was run a total of 5 times, since the result is not absolute and can change when only changing the enemies' spawn points while keeping all enemies, types and parameters in each fight. The winner of the fight is the team that wins the majority of the times.

We were then able to simulate each of the 50 fights and predict its outcome according to the calculated values. The heuristic was validated with an accuracy of 92%.

The heuristic that was implemented to simulate the fights between enemy teams is heavily designer dependent, which comes as a disadvantage, but its simplicity is also advantageous since it performs a high number of simulations in a short amount of time

and minimal resources and, in the case of new content is added, the heuristic can be adapted by the designer.

6.6 Random Challenge Model

In this model, challenges are randomly generated and selected for each of the 20 rooms.

For testing purposes, a way for all the play-testers to have the same randomly generated enemy teams selects for the 20 rooms had to be ensured. The selected challenges for this pseudorandom selection are as follows in **fig. 5**:

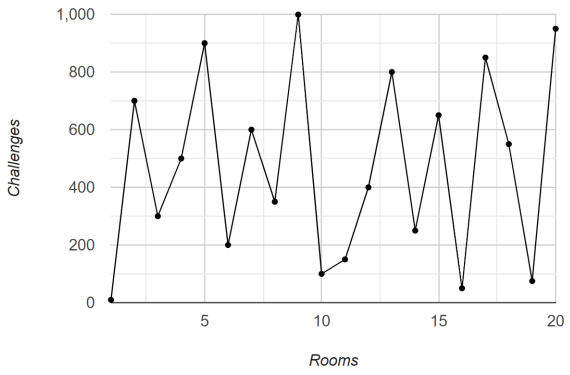

Figure 5: Pseudorandom selection for the random challenge model

In this graph, we can see the depicted difficulty line as the player moves from room to room. The higher the value in the y-axis, the more difficult the room is. This difficulty line is full of peaks and valleys, opposing the traditional lines with steadily increasing difficulty and small peaks followed by valleys in possible boss rooms. The depicted pattern was chosen in order to ensure a wide variety of difficulty changing from room to room.

6.7 Non-Adapted Challenge Model

In this model, a total of 20 challenges had to be selected, one for each room of the run, with each being more difficult to the player to complete.

Using the *.csv* file with the 1000 generated challenges, which are already ordered by difficulty, we are able to select each challenge with an increasing level of difficulty, according to the developed heuristic.

The selected challenges of increasing difficulty form a steady line with the expression $y = 50x + 50$, with y and x representing the selected challenge and the number of the room, respectively.

6.8 Adapted Challenge Model

In this model, 20 challenges have to be selected, taking into account the player's performance in the selection of the next challenge.

In order to accomplish that, the *.csv* file with the 1000 generated enemy teams is used and each is treated like a player entity from a multiplayer setting, which has a corresponding Elo value, representing their skill level and consequent difficulty level for the player.

In this work, the starting set value used was 1000, which allows obtaining values in the range of 0 to around 3000, but can be easily changed if different ranges of Elo values are desired.

Upon the first run of this model starting, there is a bootstrap. This consists of having all enemy teams undergo a certain number of fights with each other, so their Elo values shift from the initial and create a wide range of their skill levels. These fights are simulated with the heuristic predicting the outcome of each fight, significantly reducing the amount of time and computational power need to complete the fights.

In this work, the number of fights each enemy team has on the bootstrap is 20, but can be changed at will. A bigger number than this will turn the time to load this model longer, since each of the generated enemy teams will have to undergo a higher number of fights which the heuristic will have to predict, and a lower number might not create a range of Elo values wide enough, not accomplishing the objective of the bootstrap.

The process of selecting each challenge the player will face for the whole run is as follows:

- The player fights the enemy team with the closest Elo rating to them. While this is happening, each of the remaining enemy teams fights another one;
- Upon fight completion, the Elo rating of each team (player and all enemy teams) is adjusted according to the result;
- The player goes through each room fighting an enemy team with the closest Elo rating to the player's.
- As the player is progressing, if they win, the next enemy team selected will have a higher Elo rating and, if they lose, the next enemy team will have a lower Elo rating, thus merging to a certain Elo rating value corresponding to the player's skill level and corresponding challenging enemy difficulty.

6.9 Tutorial

The tutorial is composed by rooms, with information being prompted as the player goes through each room, which they have to shoot a target to open the door. This information is shown in the form of text on a sign when the player gets close to it.

In three specific rooms, enemy teams are spawned, so the player can have their first experience fighting them. In these cases, although damage indicators that the player is being hit are activated, they have unlimited lives, allowing to practice and possibly strategize against the future enemies they will be fighting.

7 EVALUATION

Besides all the data that is registered while playing, such as each run's number of points, we want to measure the players' subjective experience related to their intrinsic motivation and self-regulation. To do so, we measured 4 distinct dimensions of the IMI: Interest/Enjoyment, Perceived Competence, Effort/Importance and Pressure/Tension. We also intended to measure the player's positive and negative affect of their experience related to each of the three versions, which we measured using a smaller version of PANAS called I-PANAS-SF[Kercher 1992].

A questionnaire was prepared using Google Forms[19] where data related to the aforementioned subjective information, how the player felt the challenges were being generated and the most and least preferred version.

7.1 Pilot Evaluation

A total of 6 people participated in the preliminary evaluation.

In order for the players to not be able to identify each of the three versions, they were anonymized: the R version became version X, the NA version became version Y and the A version became version Z. To avoid having everyone trying each version in the same order, a total of 6 combinations were prepared: XYZ, XZY, YXZ, YZX, ZXY and ZYX.

The order of preference for the three versions was, from most preferred to least preferred, versions Z, Y and lastly X. The common reason for this preference on version Z was based on players feeling like this version was constantly providing challenges that seemed "difficult enough", while version X felt like it's difficult was too random. These were promising results, as these were known characteristics of versions A and R, respectively.

Players were able to identify some of the enemies and their corresponding behavior and shooting patterns. This allowed them to establish strategies such as eliminating the most dangerous enemies first, going around the room to avoid all enemy shots, grouping enemies in the center of the room (so it was easier to throw and retrieve their weapon) and reloading before moving to the next room (so they would have their weapon's clip full).

7.2 Final Evaluation

The participant starts by reading the first page of the questionnaire, where they give consent in being informed their participation is voluntary. The participant advances then to the next page, where questions related to the participant's demography and gaming habits are asked. Based on the versions order that was attributed to the player, the next section is selected.

The participant then plays the version indicated by the overseer twice and, upon completing them, answers the questionnaire section related to that version. After answering all of that version's questions, the overseer indicates the next version for the participant to play. This process is repeated until all versions are played twice and all related questionnaire sections are filled.

When the play session is completed, the participant answers the final questionnaire section, which is related to the preference regarding each of the versions and why.

7.2.1 Results and Changes.

A total of 30 participants took part on the final evaluation procedure, with ages varying from 20 to 27 years old. 19 of the participants answered they make time on their schedule to play video games.

When asked if they are familiar with games where the protagonist combats a large number of enemies by shooting at them while dodging their fire, 22 of them answered they enjoy these games and have played/watched others play them multiple times.

[19]https://www.google.com/forms/about/

With each participant playing each of the three versions twice, the number of points from a total of six runs was recorded.

A Repeated Measures One-Way ANOVA test with a Greenhouse-Geisser correction determined that there is statistically significant difference on the obtained points for each run ($F(3.794, 110.026) = 3.575$, $p = 0.01$), since p is below 0.05.

In order to find the pairs in which the statistically significant different is present, we followed with a post-hoc analysis of pairwise comparisons, in which we can see there are two pairs where the p value is below 0.05, meaning there is a statistically significant difference in pairs NA Version Run 1 - A Version Run 2 and A Version Run 1 - A Version Run 2.

This difference is justified with NA Version Run 1 being one of the runs where participants are able to get more points from trying the linearly increasing difficulty model for the first time. On the A Version, players are able to get more points in the first run from fighting enemy teams that correspond to a medium level of difficulty. Upon reaching the second run of the A Version and possibly achieving a good result in the first run, will face enemy teams with a high Elo skill rating, thus being harder to get higher scores.

Variables were used to measure the participants' subjective experience in both the 4 distinct dimensions that are appropriate to our work of the IMI and the positive and negative affect from PANAS. A variable was also used to measure how the participants felt regarding the challenge selection. These 7 measured variables are: *Average_Interest_Enjoyment*, *Average_Perceived_Competence*, *Average_Effort_Importance*, *Average_Pressure_Tension*, *Positive_Affect*, *Negative_Affect* and *Enemy_Team_Selection*.

The analysis of each of the collected 7 variables as follows:

- **Interest/Enjoyment:** Obtained value is closer to the possible maximum, meaning the participants perceived being very interested and really enjoyed playing all three versions;
- **Perceived Competence:** Obtained value is as close to the possible maximum as well as the average, meaning the participants perceived being competent playing all three versions;
- **Effort/Importance:** Obtained value is close to the possible maximum as well as the average, meaning the participants perceived to make an effort and felt these were important tasks to complete;
- **Pressure/Tension:** Obtained value is in the smaller values of the scale, tending to the average, meaning the participants perceived to not be that pressure or to feel that tense while completing the tasks;
- **Positive Affect:** Obtained value supports the idea that participants feel a strong positive affect. It also accomplishes the goal to be a higher value compared to the variable that represents the negative affect in all three versions;
- **Negative Affect:** Obtained value supports the idea that participants feel a week negative affect. It also accomplishes the goal to be a smaller value compared to the variable that represents the positive affect in all three versions;
- **Enemy Team Selection:** Obtained value from asking the participants, on a scale from 1 to 5, with 1 being "completely

random" and 5 "completely deterministic", how they felt regarding how the enemy teams were selected. Version X's value sits in the middle of the scale with a value of 2.53, while both version Y and Z approximate to the deterministic side of the scale with approximate values of 3.3.

We started by testing each of these variables for normality, using *Shapiro-Wilk*[20], in order to decide if we would use parametric or non-parametric tests.

The *Sig.* value of *Average_Interest_Enjoyment_Y*, *Negative_Affect_Y* and *Enemy_Team_Selection-Y* is below 0.05, which means the data significantly deviates from a normal distribution and non-parametric tests must be done.

In the cases where we were able to perform parametric tests on a variable for all three versions, we tested them using the One-Way ANOVA with Repeated Measures[21]. This test was performed with a Greenhouse-Geisser correction determined that there is no statistically significant difference on the perceived effort/importance between the three versions ($F(1.985, 57.560) = 0.008$, $p = 0.992$), since p is greater than 0.05.

The same conclusion was obtained when performing the test to *Average_Pressure_Tension*, *Average_Positive_Affect* and *Positive_Affect*, which determined that there is no statistically significant difference on the perceived pressure/tension ($F(1.810, 52.476) = 0.692$, $p = 0.491$), perceived positive affect ($F(1.909, 55.367) = 0.658$, $p = 0.515$) and perceived competence ($F(1.582, 45.892) = 0.796$, $p = 0.43$).

Although the *Average_Perceived_Competence* variable passed on the Shapiro-Wilk normality test with *Sig.* values greater than 0.05 for all three versions, it fails on the Mauchly's Test of Sphericity[22] ($\chi^2(2) = 8.577$, $p = 0.014$), which means p is below 0.05, thus violating sphericity.

This violation means the equivalent non-parametric test can be used to verify for statistically significant difference of this variable, in this case, the Friedman Test[23]. This is also the test that we are performing in the cases where, at least in one of the three versions, the variable requires non-parametric tests.

This test confirms the result we previously obtained regarding the *Average_Perceived_Competence* variable, which fails the Friedman Test ($\chi^2(2) = 1.982$, $p = 0.371$), since p is greater than 0.05 and thus meaning there is no statistically significant difference on the perceived competence between the three versions.

The same conclusion was obtained when performing the test to *Average_Interest_Enjoyment* and *Negative_Affect*, which determined there is no statistically significant difference on the perceived interest/enjoyment ($\chi^2(2) = 1.717$, $p = 0.424$) and negative affect ($\chi^2(2) = 3.895$, $p = 0.143$).

Upon performing the Friedman Test on the *Enemy_Team_Selection* variable, we obtained different results ($\chi^2(2) = 6.675$, $p = 0.036$), with p below 0.05, which means there is a statistical significant difference of this variable between the three versions. In order to find out in which versions this difference is statistically significant, we performed the Wilcoxon Signed-Rank Test[24].

With the obtained results from this test we concluded that there is a statistically significant difference of *Enemy_Team_Selection* between versions X and Y ($Z = -2.486$, $p = 0.013$), there is a statistically significant difference of *Enemy_Team_Selection* between versions X and Z ($Z = -2.500$, $p = 0.012$) and there is not a statistically significant difference of *Enemy_Team_Selection* between versions Y and Z ($Z = -0.287$, $p = 0.774$). Applying the Bonferroni correction, only in cases where p is below 0.17 (0.05 / 3) we obtain a statistically significant difference. This means that the values obtained of *Enemy_Team_Selection* from version X are significantly different from those obtained from versions Y and Z.

Lastly, the preference of each participant regarding what was their most and least preferred version was registered from the last page of the questionnaire. The bar graph that shows the participants' preference can be seen in **fig. 6**.

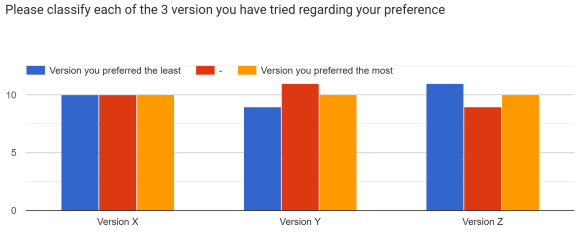

Figure 6: Graph bar of the participants' most and least preferred versions

In the depicted figure, we can see that each of the three versions had the same number of votes as "Most preferred version", with a total of 10 votes each, represented by the orange bar. More people classified version Z as the "Least preferred" version, with one less vote than version X and two fewer votes than version Y, represented by the blue bar.

7.3 Discussion

The first step for testing the stated hypothesis is to look at the bar graph on **fig. 6**, where each participant chose which version they preferred the most and the least. This graph does not allow us to take objective conclusions, since the results are too similar, not having a significant difference to allow us to classify a version as being more or less preferred than the others.

The measured subjective experience might allow us to draw conclusions, since in this case the obtained data is a result from how they perceived their experience, which might be different to the player's objective preference.

Upon performing the tests in order to verify if there is a statistically significant difference between the obtained results, we conclude this difference does not exist in any of the six aforementioned variables, which means there is no difference in perceived experience between the three versions.

[20] https://statistics.laerd.com/spss-tutorials/testing-for-normality-using-spss-statistics.php
[21] https://statistics.laerd.com/spss-tutorials/one-way-anova-repeated-measures-using-spss-statistics.php
[22] https://statistics.laerd.com/statistical-guides/sphericity-statistical-guide.php
[23] https://statistics.laerd.com/spss-tutorials/testing-for-normality-using-spss-statistics.php
[24] https://statistics.laerd.com/spss-tutorials/wilcoxon-signed-rank-test-using-spss-statistics.php

There is a statistically significant difference on the seventh tested variable, which relates to how the players perceive the challenges to be selected to each room. There is a perception that in version X, corresponding to the R version, enemies are closer to being randomly selected than in versions Y and Z, corresponding to NA and A versions, respectively, which are closer to being deterministically selected. This means the players were able to distinguish the model where difficulty is adapted to their performance to the model where difficulty, but not from the model where difficulty increases linearly. This last distinction might have been cleared with an increased experience time of these two models.

8 CONCLUSIONS

In our work, we present an approach to a PCG single-player video game, where the difficulty of the challenges is adapted to the player's performance and compare it to versions of the same game where difficulty follows a linear increment or changes randomly.

Since we knew that different players have different playing experience and hence different skill levels, we intended to create a model that would provide the challenge which difficulty would match the player's skill throughout the whole play session.

Three versions were developed, which used Holiday Knight, a *Bullet Hell Dungeon Crawler* as a testbed game, in which the player has to clear the room of all enemies in order to progress to the following challenges.

Each version differs from the others in the selection process of the challenge for each room. One version takes into account the player's performance, the other keeps increasing the challenge's difficulty and the last is random. Our hypothesis was that players would prefer the first version the most, followed by the second and, lastly, the third.

This hypothesis was tested by having a total of 30 participants trying each of the three versions twice, providing feedback, answering questions related to the versions and by choosing the preferred one. The points from all the player's runs were also recorded.

The final evaluation showed that there was no statistically significant difference between the preferred version chosen by the players, neither in how players perceived their experience to be. There was, however, a statistically significant difference between how the players felt the challenges were selected in each version, saying that the selection from the version where challenges are randomly selected were indeed random, while in the other two versions, the selection was deterministic, in other words, there was a reason for that specific challenge to have been selected, as, although this selection process is different between them, it is a deterministic process.

The fact there was no difference on the participants' preferred version and no statistically significant difference on the perceived experience between the three versions does not allow us to possibly confirm or refute our stated hypothesis.

REFERENCES

Maria-Virginia Aponte, Guillaume Levieux, and Stephane Natkin. 2011. Measuring the level of difficulty in single player video games. *Entertainment Computing* 2, 4 (2011), 205–213.

Francisco Bicho and Carlos Martinho. 2018. Multi-Dimensional Player Skill Progression Modelling for Procedural Content Generation. In *Proceedings of the 13th International Conference on the Foundations of Digital Games* (Malmö, Sweden) *(FDG '18)*. Association for Computing Machinery, New York, NY, USA, Article 1, 10 pages. https://doi.org/10.1145/3235765.3235774

Francisco Fino Nina Pina Bicho. 2018. *Multi-dimensional Player Skill Progression Modeling for Procedural Content Generation.* Master's thesis. Instituto Superior Técnico.

Matthieu JS Brinkhuis and Gunter Maris. 2009. Dynamic parameter estimation in student monitoring systems. *Measurement and Research Department Reports (Rep. No. 2009-1). Arnhem: Cito* (2009).

Cameron Browne and Frederic Maire. 2010. Evolutionary game design. *IEEE Transactions on Computational Intelligence and AI in Games* 2, 1 (2010), 1–16.

Robin Hunicke. 2005. The case for dynamic difficulty adjustment in games. In *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology*. 429–433.

Kyle Kercher. 1992. Assessing Subjective Well-Being in the Old-Old: The PANAS as a Measure of Orthogonal Dimensions of Positive and Negative Affect. *Research on Aging* 14, 2 (1992), 131–168. https://doi.org/10.1177/0164027592142001

Nicholas E Newton-Fisher. 2017. Modeling social dominance: Elo-ratings, prior history, and the intensity of aggression. *International journal of primatology* 38, 3 (2017), 427–447.

João Filipe Lopes Pardal. 2019. *Holiday Knight: a Videogame with Skill-based Challenge Generation.* Master's thesis. Instituto Superior Técnico.

Radek Pelánek. 2016. Applications of the Elo rating system in adaptive educational systems. *Computers & Education* 98 (2016), 169–179.

Noor Shaker, Julian Togelius, and Mark J Nelson. 2016. *Procedural content generation in games.* Springer, Chapter Introduction, 1–15.

Gillian Smith. 2014. Understanding procedural content generation: a design-centric analysis of the role of PCG in games. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 917–926.

Colin SC Tsang, Henry YT Ngan, and Grantham KH Pang. 2016. Fabric inspection based on the Elo rating method. *Pattern Recognition* 51 (2016), 378–394.