Modelling Player Skill in a Procedurally Generated Single-Player Videogame

Jorge Humberto Costa Nunes

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisor: Prof. Carlos António Roque Martinho

November 2022

This work was created using LaTeX typesetting language
in the Overleaf environment (www.overleaf.com).

Acknowledgments

I would like to start by thanking my parents. Your friendship, encouragement and caring got me where I am today and that would not be possible without your support. You managed to deal with me through my ups and downs and were always there for me through thick and thin. I also want to thank my family, whose encouragement words got me going even through the toughest moments.

I would also like to thank my friends. Every single one of you. You know who you are, so there is no reason so single out any of you. You got me laughing whenever I felt like crying and encouraged me to get up every day, so we could work together, remotely or in person. I will not forget our moments, our calls, our encounters, our games.

Thank you to everyone who took a bit of their time to test my project, really hope you enjoyed trying it and hope to have inspired some of you who might be looking forward to working in the game industry.

Last but not least, I would also like to acknowledge my dissertation supervisor Prof. Carlos Martinho for your insight, support, sharing of knowledge and, most of all, patience to put up with me. You made this Thesis possible.

To each and every one of you – Thank you.

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 dissertation, 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.

Keywords

Flow; Single-Player; Procedural Content Generation; Dynamic Difficulty Adjustment; Player Skill; Subjective Experience Measurement

Resumo

A dificuldade é um dos maiores incentivos nos jogos individuais, daí a sua importância em como avaliar a habilidade do jogador e o escalar da dificuldade, enquanto o jogador ganha experiência e mantém o seu estado de envolvimento.

Nesta dissertação, exploramos como um sistema Elo pode ser usado para criar um sistema de Ajuste Dinâmico de Dificuldade num jogo *bullet-hell* individual gerado procedimentalmente, chamado *Holiday Knight*.

A nossa abordagem pretende testar se o jogador se diverte mais ao jogar um jogo em que há geração procedimental enquanto a sua dificuldade é adaptada conforme a sua habilidade enquanto joga, comparado a uma versão do mesmo jogo em que a dificuldade aumenta linearmente e a uma versão em que os desafios são selecionados aleatoriamente.

Na nossa avaliação, os participantes (N = 30) jogaram o jogo e responderam a um questionário, cujo foco é medir a sua motivação intrínseca, a sua afetividade percecionada e que versão preferiram.

Não foi possível encontrar uma diferença estatística significativa que possa apoiar qual foi a versão preferida pelos jogadores, mas conseguiram identificar a aleatoriedade da versão em que os desafios eram de facto selecionados aleatoriamente e o determinismo das versões em que a dificuldade dos desafios é adaptada ao desempenho do jogador e da versão em que a dificuldade aumenta linearmente.

Palavras Chave

Envolvimento; Jogador Individual; Geração Procedimental de Conteúdo; Ajuste Dinâmico de Dificuldade; Habilidade do Jogador; Medição Subjetiva de Experiência

Contents

List of Figures

List of Tables

Acronyms

ANN	Artifical Neural Networks
DDA	Dynamic Difficulty Adjustment
GEQ	Game Experience Questionnaire
IMI	Intrinsic Motivation Inventory
PANAS	Positive and Negative Affect Schedule
PCG	Procedural Content Generation
PVP	Player Versus Player

1 Introduction

Contents

1.1 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 [1].

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 slow down, hitting more obstacles and generally not doing as well, if the player is struggling, but, on the other end, if the player is doing extremely well, the AI will give the CPU racers a bit of a speed boost, making it so they are always at least threatening to get back in the race[9].

[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/

1.2 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.

As later described in 2.4, 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.

1.3 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 [2], 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.** 1.1 and further described in chapter 3), 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

[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.

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.

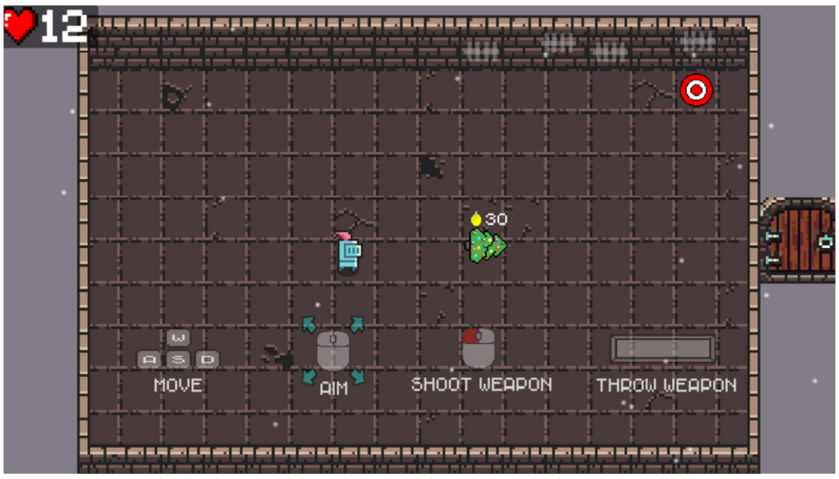


Figure 1.1: Holiday Knight's tutorial room

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.

1.4 Objectives

This body of work starts with the study of flow, which will be the basis of our research.

We examine the state of art and related work of Procedural Content Generation, Dynamic Difficulty Adjustment (DDA) and the player's modeled skill and progression, in order to provide further comprehension on the mentioned topics.

We also propose a way to model multiplayer skill level matchmaking systems into a single player game environment.

With this work, we also adapt an existing video game where procedural content generation occurs and develop three versions of this build: one where the difficulty of the enemies faced is random, one where difficulty keeps increasing linearly and another where each battle's difficulty is dynamically adjusted by the implemented skill level matchmaking system.

Lastly, after having user tests, we study the results, feedback and measured subjective experience from all implemented video game version and compare them through statistical analysis in order to draw conclusion and to be possible to verify our hypothesis.

1.5 Document Outline

This thesis is organized as follows:

- In Chapter 1 we present the **Introduction**, where we discuss the motivation that drove this work to happen (1.1), we state the problem that it arouse (1.2) and prompted hypothesis, which our should confirm (1.3). Finally, we declare the objectives of the work itself (1.4).

- In Chapter 2 we present the **Related Work**, where we focus on multiple important topics that gave this work a direction such as the definition of Cognitive Flow (2.1), what is Procedural Content Generation, how it is used in video games, its types, properties and taxonomy (2.2), the role of difficulty as one of the main core of video games and concepts such as Static Difficulty and Dynamically Adjusted Difficulty (2.3), we study Player Skill, Skill Atoms and Progression Models depicted in previous work and Statistical Models that are commonly used in video games (2.4) and finally we approach the topic of Subjective Experience Measurement (2.5) to obtain and analyze data which is related to each participant's own experience.

- In Chapter 3 we present the needs of the **Testbed Game**, so we can make a Choice of the Testbed Game to test our hypothesis (3.1) and the characteristics of the chosen game, Holiday Knight (3.2). We follow up with the possible Character Actions that can be performed in the game (3.3) and what consists on a Game Loop of our developed version of Holiday Knight (3.4). Finally, we delve in the created Game Environment, such as what composes a room and the characteristics of the enemies (3.5);

- In Chapter 4 we present the **Implementation** of the developed version of Holiday Knight. We start with the concept of Dynamic Challenge Adaptation (4.1) and the advantages it brings to our work, followed by how Challenge Generation is performed (4.2). We then describe the creation of the Battleground Scene (4.3), which will support the implemented Challenge Selection Heuristic that classifies the challenges' difficulty (4.4). After being developed, we present the Heuristic Validation process (4.5) so that the use of this heuristic can be supported. We then delve into the PCG properties and taxonomy (4.6) of the three developed models: the Random Challenge Model (4.7), the Non-Adapted Challenge Model (4.8) and the Adapted Challenge Model (4.9). Lastly, we describe the Tutorial (4.10), which presents the player to Holiday Knight's mechanics;

- In Chapter 5 we present the **Evaluation** process which will allow us to confirm or reject the stated hypothesis. We start by describing the Preliminary Evaluation (5.1), such as its procedure and the obtained results and consequent changes that had to be made. We follow up with the Final Evaluation (5.2.1), starting with its procedure and ending with the obtained results, which are then organized and analyzed. We finish with the Discussion (5.3), where we discuss the possibilities, origins and possible conclusions that can be drawn with the obtained results;
- In Chapter 6 we present the **Conclusions** that were made regarding our work (6.1) and possible Future Work that may be based on the implemented project and gathered results (6.2).

2

Related Work

Contents

In this section we will be presenting topics such as the Cognitive Flow, Procedural Content Generation, Difficulty, Player Skill, Skill Atoms, Progression Models and Subjective Experience Measurements.

2.1 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[1].

While in the flow state, more commonly known as "being in the zone", people experience:

- Extreme focus on a task;
- A sense of active control;
- Merging of action and awareness;
- Loss of self-awareness;
- Distortion of the experience of time;
- The experience of the task being the only necessary justification for continuing it.[2]

Psychologist Mihály Csikszentmihalyi [3] proposed the concept, which has been widely referred to flow state across a variety of fields. Csikszentmihalyi also outlined four characteristics found in tasks that drive an equilibrium between skill and difficulty, thus increasing the probability of flow states. The following tasks should be considered by game developers if they want to increase the likelihood of causing flow states in the players:

- Have concrete goals with manageable rules - in moments where players don't know what to accomplish and what techniques are usable to solve a puzzle, player's would turn frustrated;
- Demand actions to achieve goals that fit within the person's capabilities - if a player is unable to accomplish their goal because they feel they are inept, like flawlessly cutting through a fast barrage of blocks in Beat Saber[3];
- Have clear and timely feedback on performance and goal accomplishment - without some kind of feedback on the right moment, there is no way for the player to know how they are doing;

[1] https://positivepsychology.com/what-is-flow/
[2] https://www.gamedeveloper.com/design/cognitive%2Dflow%2Dthe%2Dpsychology%2Dof%2Dgreat%2Dgame%2Ddesign
[3] Beat Saber (Oculus Quest, PC, PlayStation 4, 2019), a virtual reality rhythm video game, developed and published by Beat Games.

- Diminish extraneous distraction, thus facilitating concentration - an exaggerated HUD on a FPS video game, would take away the attention from the player from what is happening[4].

The measurement of how engaged a person is while performing a certain task depends on the relation of how challenging that task is and the level of skill of the person performing it. The relation between these dimensions may be seen in **fig.** 2.1.

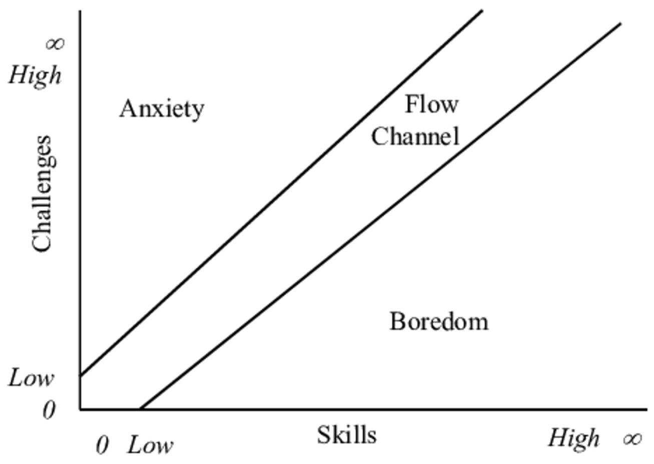


Figure 2.1: Flow Channel according to Csikszentmihalyi

In the mentioned figure, 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.

In order for the player to stay engaged to the game and stay in the Flow Channel, the level of the challenge must increase as the player's skills do too. An easier challenge should be given when the player is still learning the mechanics of the game and what techniques they can use (in the tutorial or early stages), while the hardest challenges should be provided when the player is mastering the game.

In their work [4], Chanel et al., by having a variety of players playing a Tetris game at different difficulty levels and accurately detecting their physiological signals, they were able to conclude that the engagement of a player can decrease if the game difficulty does not change. This confirms that without changing the challenge level while the player's skill is increasing, will leave them in the boredom area, as depicted on **fig.** 2.1.

[4]Check footnote 16.

Finally, in his work [5], depicted on **fig. 2.2**, Hanin studied how levels of stress affected the performance of athletes in sports. Peak performance was only achieved with the right amount of arousal/stress. When this psychological factor was either too high or too low, the athlete's performance pummeled down. Although being directed to sports, Hanin's work also supports Csikszentmihalyi model of the Flow Channel: the right amount of arousal/stress will be obtained from the right level of a challenge, which will get the most engagement (and consequent focus and possible performance) from a player with the proportional skill level.

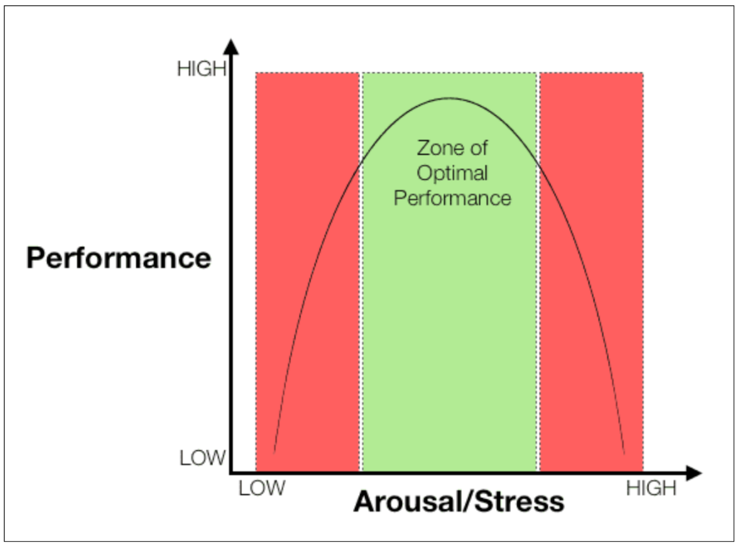


Figure 2.2: Performance Function according to Hanin

2.2 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[5]. 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 only ones that were prepared by the designers themselves, thus increasing game replayability.

Togelius et al. [6] 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. Instead of spending possibly years creating huge open-worlds and each of their little corners, designers are now able to do so 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 [7] regarding evolutionary game design, where they study both a system's ability to automatically measure generated

[5] http://pcg.wikidot.com/what-pcg-is

games and their potential to interest human players, and its ability to create new high-quality games.

Another work that closely follows the capabilities of PCG is Tanagra [8]: the human designer is able to place constraints on a continuously running level generator, in the form of exact geometry placement and manipulation of the level's pacing. This allows the designer to generate a 2D level using a constraint satisfaction algorithm to fill any undesigned open spaces, while keeping the game in a playable state or, in other words, able to be played and successfully completed by a human. **fig.** 2.3 depicts the level design tool that was created for this work.

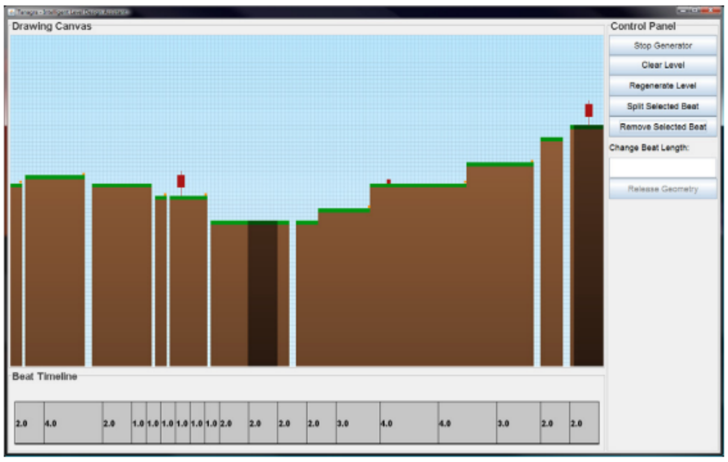


Figure 2.3: The Tanagra intelligent level design tool

2.2.1 PCG Classification

Barriga [9] classifies game content that can be procedurally generated in the following way:

- **Game Bits** - The fundamental unit of game content. They can be necessary, such as the texture to differentiate the main character from the enemies, or optional, such as background music, but either way, they usually do not interact with the player directly when considered independently. Game bits include textures, sound, vegetation, architecture and graphic effects (fire, water, smoke, etc.);

- **Game Space** - The environment in which the game takes place, and includes maps and terrain;

- **Game Systems** - These are responsible for simulating more complex environments such as ecosystems, road networks, urban environments and entity interactions;

- **Game Scenarios** - They organize the previous levels into coherent plans or sequences of events. They include puzzles, stories and levels;

- **Game Design** - The set of rules and mechanics of the game;
- **Derived Content** - Content that can be created as a companion to the game world, like leaderboards and news items covering player actions in the game world.

2.2.2 The Types of PCG

The article *The Death of the Level Designer* [10] outlines seven types of procedural content generation, with respect to games:

- **Runtime Random Level Generation** - In this case, game scenarios, more specifically, game levels, are generated while the game is being played. This is the most envisaged type of PCG and was originated by Rogue[6], which randomly places rooms and corridors when a player first enters the game level. This type of PCG provides an infinite game space, constrained by the generation rules of the used algorithm;
- **Design of Level Content** - Uses PCG techniques at the design stage to build game assets such as levels, enemies and weapons. Due to the increasing cost of content design, PCG can supplement human design skills to rapidly populate the environment with randomly generated content, that can then be altered by hand to fit the intended design. Alternately, by rapidly iterating through a number of PCG generated levels, the designer can select a level that closest fits the game requirements. A classic example of Design of Level Content in use is generating a heightmap[7] using PCG techniques such as fractals[8] or Perlin Noise[9] which then has hand-placed content added to it;
- **Dynamic World Generation** - Uses a random seed or hash of a user supplied string which remains constant, and then grow the playing field iteratively using this number permuted by pseudorandom number generator techniques. One of the first examples of a game that achieves dynamic world generation is Elite[10]. There are several issues to be aware of with this method. The first is that changing the algorithm, changes the world. What this means is that tweaking any parameters or techniques in the procedural content generation method will completely change the generated content;

[6]Rogue (Amiga, Amstrad CPC, Atari 8-bit, Atari ST, Commodore 64, CP/M, DOS, Macintosh, TOPS-20, TRS-80 CoCo, Unix, ZX Spectrum, 1980), a dungeon crawling video game, developed by Michael Toy and Glenn Wichman with later contributions by Ken Arnold and published by Epyx.

[7]a system that stores terrain by representing height (Z) as a function of horizontal position (X,Y), that is Z = f(X,Y).

[8]a description of a broad set of shapes.

[9]a procedural generation algorithm used to generate things like textures and terrain.

[10]Elite (BBC Micro, Acorn Electron, Apple II, Amstrad CPC, Commodore 64, ZX Spectrum, MSX, Tatung Einstein, IBM PC compatible, Acorn Archimedes, Amiga, Atari ST, Nintendo Entertainment System, 1984), a space trading and combat simulator video game, developed by David Braben and Ian Bell and published by Acornsoft and Firebird.

- **Instancing of In-Game Entities** - An increasingly common technique, particularly in middleware such as SpeedTree[11], is to dynamically vary the parameters of in-game entities to create a large possible number of entities with a statistically insignificant chance of repetition. This could be the visible geometry of a tree or person, or the in-game properties of a weapon or piece of equipment. Various PCG techniques such as pseudorandom number generation can be used to instance the entities, while compressing the total amount of information required to be held by the game for each unique entity;
- **User Mediated Content** - This method harnesses the strengths of human perception and selection criteria, with the strengths of computer processing power and procedural computation and can take advantage of distributed intelligence such as used in the ESP Game[12] to make the most favored content widely available. Spore[13] uses this type of PCG in order to generate all its in-game content;
- **Dynamic Systems** - Systems such as weather, and group and crowd behavior, can be modelled using PCG techniques. AI and PCG usually overlap in this area, with AI systems such as the Radiant AI[14] in The Elder Scrolls IV: Oblivion[15] and A Life system in S.T.A.L.K.E.R.: The Shadow of Chernobyl[16] causing individuals to respond dynamically to game situations;
- **Procedural Puzzles and Plot Generation** - At the simplest level, procedural content generation can be used to change the door codes and other individual puzzle elements to prevent a user from getting the information off of a game FAQ website or other source of information. But many game plots are little more than dependency graphs, in which a sequence of actions must be performed in a particular order. Puzzles can be extended by making multiple parts of the dependency graph randomly placed (e.g. moving the key that opens the door to a random accessible location) or by changing the shape of the dependency graph completely.

2.2.3 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. [6] states a list of common desirable properties of PCG solutions:

[11]https://store.speedtree.com/

[12]The idea behind the game is to use the computational power of humans to perform a task that computers cannot (originally, image recognition) by packaging the task as a game.

[13]Spore(PC, 2008), a life simulation real-time strategy God video game, developed by Maxis and published by Electronic Arts.

[14]https://elderscrolls.fandom.com/wiki/Radiant_A.I.

[15]The Elder Scrolls IV: Oblivion (PC, PlayStation 3, Xbox 360, 2006) is an open-world action role-playing video game, developed by Bethesda Game Studios and published by Bethesda Softworks and 2K Games.

[16]S.T.A.L.K.E.R.: The Shadow of Chernobyl (PC, 2007) is a first-person shooter survival horror video game, developed by GSC Game World and published by THQ and GSC World Publishing.

- **Speed** - Requirements for speed vary wildly, from a maximum generation time of milliseconds to months, depending on (amongst other things) whether the content generation is done during gameplay or during development of the game;
- **Reliability** - Some generators "shoot from the hip", whereas others are capable of guaranteeing that the content they generate satisfies some given quality criteria. This is more important for some types of content than others, for example a dungeon with no exit or entrance is a catastrophic failure, whereas a flower that looks a bit weird just looks a bit weird without this necessarily breaking the game;
- **Controllability** - There is frequently a need for content generators to be controllable in some sense, so that a human user or an algorithm (such as a player adaptive mechanism) can specify some aspects of the content to be generated. There are many possible dimensions of control, e.g. one might ask for a smooth oblong rock, a car that can take sharp bends and has multiple colors, a level that induces a sense of mystery and rewards perfectionists, or a small ruleset where chance plays no part;
- **Expressivity and Diversity** - There is often a need to generate a diverse set of content, to avoid the content looking like it's all minor variations on a tired theme. At an extreme of non-expressivity, consider a level "generator" that always outputs the same level but randomly changes the color of a single stone in the middle of the level; at the other extreme, consider a "level" generator that assembles components completely randomly, yielding senseless and unplayable levels. Measuring expressivity is a non-trivial topic in its own right, and designing level generators that generate diverse content without compromising on quality is even less trivial;
- **Creativity and Believability** - In most cases, we would like our content not to look like it has been designed by a procedural content generator. There are a number of ways in which generated content can look generated as opposed to human created.

2.2.4 The Taxonomy of PCG

With the wide range of content generation challenges and solutions currently accessible, having a structure that can emphasize the differences and similarities between approaches is beneficial. Following, we present a revised number of dimensions, originally presented by Togelius et al. [11], where a single solution should be interpreted as lying between both ends of that dimension [6]:

- **Online vs Offline** - PCG techniques can be used to generate content either online while the player is playing the game, allowing for infinite variations, making the game infinitely replayable and allowing for the creation of player-adapted content, or offline during the game's development or before the start of a game session. Left 4 Dead[17] is an example of online content generation, as it provides a dynamic experience for each player by analyzing their behavior on the fly and altering the game state accordingly;

- **Necessary vs Optional** - PCG can be used to generate necessary game content that is required for the completion of a level, or it can be used to generate auxiliary content that can be discarded or exchanged for other content. The main distinctive feature between necessary and optional content is that necessary content should always be correct, while this condition does not hold for optional content;

- **Degree and Dimensions of Control** - The use of a random seed is one way to gain control over the generation space; another way is to use a set of parameters that control the content generation along a number of dimensions. In Minecraft[18], worlds are generated upon using random seeds, which means using the same seed twice will regenerate the same world;

- **Generic vs Adaptive** - Generic content generation refers to the paradigm of PCG where content is generated without taking player behavior into account, as opposed to adaptive, personalized or player-centered content generation where player interaction with the game is analyzed and content is created based on a player's previous behavior. Most commercial games tackle PCG in a generic way, while adaptive PCG has been receiving increasing attention in academia recently. A more extensive review of PCG for player-adaptive games was done by Yannakakis and Togelius [12], where they introduce a framework driven by computational models of user experience, which they name Experience Driven PCG. This model proposes dynamic content generation that optimizes the experience for the player accordingly to maximize efficacy, performance and robustness;

- **Stochastic vs Deterministic** - Deterministic PCG allows the regeneration of the same content given the same starting point and method parameters, as opposed to stochastic PCG, where recreating the same content is usually not possible. In Elite, mentioned in 2.2.2, the galaxies are regenerated in a deterministic way;

- **Constructive vs Generate-and-Test** - In constructive PCG, content is generated in one pass, which is commonly done in roguelike games. Generate-and-test PCG techniques, on the other hand, alternate generating and testing in a loop, repeating until a satisfactory solution is generated;

[17]Left 4 Dead (PC, Xbox 360, 2008) is a multiplayer survival horror video game, developed by Valve South and published by Valve.

[18]Minecraft (Windows, macOS, Linux, Android, iOS, 2011; Xbox 360, 2012; PlayStation 3, 2013; PlayStation 4, PlayStation Vita, Xbox One, 2014; Wii U, 2015; New 3DS, Nintendo Switch, 2017) is a sandbox video game, developed by Mojang Studios and published by Mojang Studios, Xbox Game Studios and Sony Interactive Entertainment.

- **Automatic Generation vs Mixed Authorship** - Until recently, PCG has allowed limited input from game designers, who usually tweak the algorithm parameters to control and guide content generation, which creates infinite variations of playable content. A new and interesting paradigm has emerged that focuses on incorporating designer and/or player input through the design process. In this mixed-authorship initiative paradigm, a human designer or player cooperates with the algorithm to generate the desired content. Tanagra, mentioned in **2.2**, is an example of this mixed-authorship PCG system.

2.3 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.

2.3.1 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, as depicted in **fig.** 2.4. In this case, the player must adapt and face the video game's difficulty in order to progress.

In their work, Aponte et al. [13] study the choice of challenges and how their succession is related to the flow principle (see section **2.1**). 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.

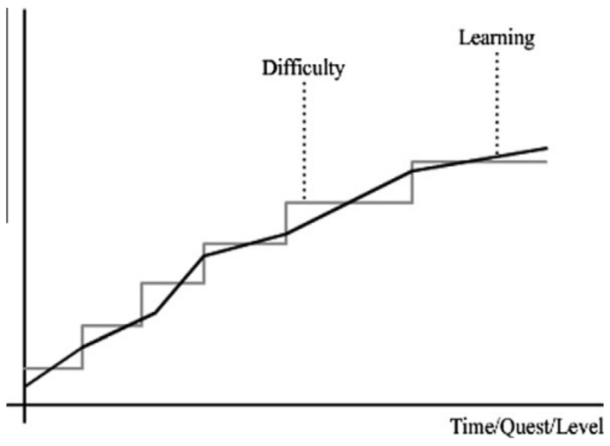


Figure 2.4: Difficulty and learning curves through time

Sarkar and Cooper [14] also study difficulty curves and attempt to modify one of them by modeling it as a function composition. Their experiment revealed that the transformed curves impact gameplay significantly and that some of them improve player engagement compared to the studied game's existing difficulty curve.

On another point, Horn et al. [15] analyzed a model player learning in a puzzle game with AI assistance. 4 different players were created, each with different capacity algorithms, in order to complete a set of puzzles. The difficulty of those problems was then categorized by noting which artificial players could solve them, with puzzles that could only be solved by the most complicated algorithm designated the most difficult and puzzles that could be solved by all algorithms labeled the easiest. The idea was to see if these computer players could mimic the difficulty that human players confront when playing this puzzle game. They did: it was discovered that categorizing the problems' difficulties by labeling which algorithms could solve them was connected to the difficulty that players experienced when solving those same puzzles (the more unsuccessful tries in a puzzle, the more difficult the challenge for the players). This demonstrates that artificial players may be used to classify a game's complexity (at least in some sorts of games).

Lastly, Allart et al. [16] study the link between difficulty and motivation in video games, using telemetric data from two games developed by Ubisoft: Rayman Legends[19] and Tom Clancy's The Division[20]. They defined difficulty as a probability of failure, estimated using a mixed effect logistic regression, and estimated the player's motivation by considering that when their motivation was too low, they would stop playing. They concluded that difficulty is by itself an explainable variable of player retention and that players tend to prefer higher levels of difficulty. Flow state does not state the optimal level of difficulty,

[19]Rayman Legends (PC, PlayStation 3, PlayStation Vita, Wii U, Xbox 360, 2013; PlayStation 4, Xbox One, 2014; Nintendo Switch, 2017; Stadia, 2021) is a platform video game, developed by Ubisoft Montpellier and published by Ubisoft.

[20]Tom Clancy's The Division (PC, PlayStation 4, Xbox One, 2016) is an online-only action role-playing video game, developed by Massive Entertainment and published by Ubisoft.

but in both games, probability of failure was almost always under what they called a balanced difficulty at 50% chances of failure.

2.3.2 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 [17] 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.

On a last note, Ang and Mitchel [18] studied the effects of DDA systems on player experience through 3 conditions: no DDA, system-oriented ramping DDA (rDDA), which is DDA that follows set system rules, and player-oriented DDA (pDDA), which is DDA oriented to the player performance. This was done with a developed minimal version of Tetris[21] Their main findings were that DDA systems do alter the game experience positively, several dimensions of what constitutes flow were demonstrated to be worse off in the absence of DDA; rDDA can impact the game experience negatively, since players have a significantly diminished sense of control; pDDA systems, on the other hand, give players a better sense of control. This study concluded with the suggestion that designers should consider DDA systems as a viable strategy to bolster experience through the manipulation of difficulty based on certain indicators, of which game performance was used in this study.

2.4 Player Skill and Progression Models

The player, their skill, and how they progress will be the major emphasis of this section. We'll look at several skill measurements and their benefits and drawbacks, as well as how skill evolves and the best approach to keep track of that progression.

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

[21]Tetris is a puzzle video game created by Soviet software engineer Alexey Pajitnov in 1984.

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.

This further supports that games with static difficulty, or as discussed in 2.3, games where difficulty progress without taking player's skill into account, might push away certain players. Dark Souls [22], which the tutorial boss can be seen in **fig.** 2.5, is widely known as being one of the hardest video games in the world. In this game, when the character that the player is controlling, the character dies, losing all progress since the last checkpoint. But this is a key part of the design of the game: death as education. This is, of course, not conventional game design. Conventional game design eases players into an experience, gradually introducing new concepts and abilities before putting you in any real danger, rather than dropping you right into a world full of things that are trying to kill you as swiftly and horribly as possible and watching you get on with it. This type of game design relies upon trusting the player to persevere, to learn from dying and try again, rather than just put down the controller and walk away, and it's baked into every design decision in Dark Souls[23].

Figure 2.5: Player facing Dark Souls' tutorial boss

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.

[22]Dark Souls is an action role-playing video game series, FromSoftware and published by Namco Bandai Games.
[23]https://www.eurogamer.net/articles/2019-12-07-tough-love-on-dark-souls-difficulty

2.4.1 Player Model

In his work, Cook[24] 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. Some skills are conceptual, such as navigating a map while others are quite physical, such as pounding in a nail with a hammer;
- **Drive to Learn** - Playing is part of the human instinct: when in a safe environment, people will begin playing by default. Strong feedback mechanisms in the form of boredom or frustration prod us into action. As game designers, we deal with fun, boredom and frustration on a regular basis, and we should recognize that these are biological phenomena;
- **Perceived Value** - Players pursue skills with high perceived value over skills with low perceived value. Perception of value is more important than an objective measurement value. Humans are not creatures of pure logic. We know people exhibit consistent biases in how they weight their actions. This is the reason people undertake bizarre risks because they are unable to properly evaluate statistical odds.

2.4.2 Skill Atoms

Still in his work, Cook defines that the player interaction with the game and how new skills are obtained can be described in a self-contained atomic feedback loop called skill atom. This loop is composed of four main elements:

- **Action** - The action performed by the player. More advanced atoms might require the player to execute a set of actions, like navigating a complex maze;
- **Simulation** - Based off the action, the simulation is updated. This might be something like a door closing or a chest that is being opened;
- **Feedback** - Feedback is provided to the player for them to know the simulation has occurred. This can be auditory, visual or tactile. An example of a feedback mechanism is the muzzle flash from a gun that has been fired;

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

- **Modeling** - The player absorbs the feedback and updates their mental model upon conclusion of the action. If they made progress or mastered a new skill, they will feel pleasure. If they fail or feel that the action has been in vain, they will feel boredom or frustration.

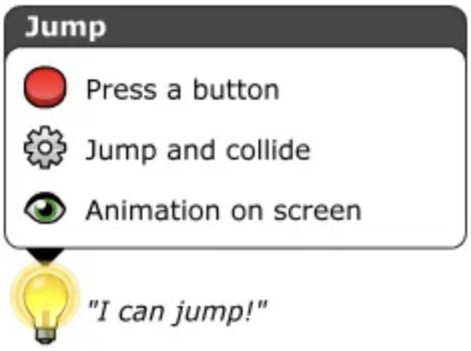


Figure 2.6: The skill atom of the player learning how to make a character jump

In **fig.** 2.6 we can see an example of a skill atom of the player learning out to make a character jump: the player performs an action, the press of a button, updating the simulation of the character moving up and then down, which results in feedback with the animation as a visual cue and, lastly, the player forms a mental model that that press of a button results in the character jumping.

2.4.3 Skill-Based Progression Model

There are multiple ways to define what's the level of a player's skill in a video game, how it is measured and how it progresses. We will include a skill-based progression model developed by Catarino in his thesis [19] and later used by Pardal [2] in his. We will also include Bicho's [20] model presented in his thesis. Finally, we will present the two most used statistical models used in games to infer skill: Elo and Glicko. As part of the Elo section, we will also present Mestre's [21] model used in his thesis.

2.4.3.A Catarino and Pardal's Progression Model

In Catarino's work [19] and article [22], a skill progression model was implemented on top of the Arena mode of Smash Time[25], where players must tap over the enemies a certain amount of times, while avoiding friendly animals.

This model's objective is to provide an engaging gameplay experience that would make players play the game more often and for longer periods. In order to do so, three different concepts were worked

[25]Smash Time (PC, 2014), is a endless clicker single player video game, developed and published by Bica Studios.

with: *Challenges*, *Obstacles* and *Tags*. These were used by the three main components that composed the core of the developed progression model: the *Player Performance Model*, the *Content Variety Model* and the *Content Generation Model*.

The *Player Performance Model* was designed to evaluate the player's skill level, in order to allow and support the game to generate adequate content that keeps constantly challenging the player. The *Content Variety Model* controls the variety of each challenge, which it represents through its novelty value and is defined by the variety of its tags. Lastly, the *Content Generation Model* used both the aforementioned models in order to generate engaging and challenging game content throughout each game session.

In this context, a challenge is a formation of obstacles with one or more paths that guide the movement of the obstacles, which can be any of the enemies the player may face. Tags are purposed to provide a way to assign, each given challenge to the player, a performance value and a variety value.

In **fig.** 2.7, we can see a representation of Catarino's work game cycle with the progression model architecture.

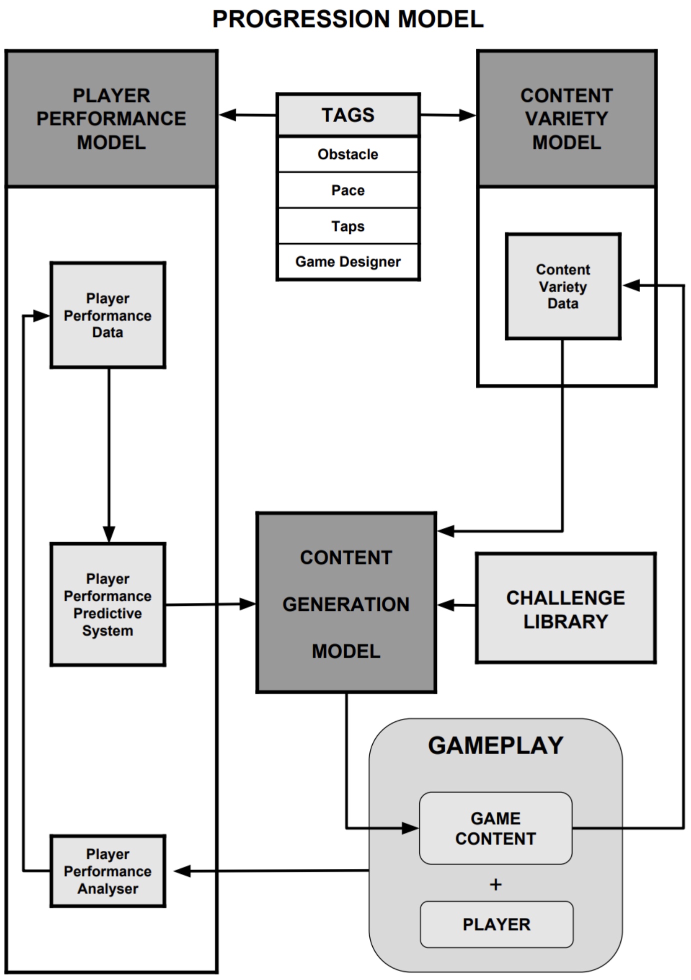


Figure 2.7: Catarino's work game cycle with the Progression Model Architecture

Performance of each tag is a variable with a value ranging from 0 to 1, which is measured individually in each challenge. Player performance on that challenge is then calculated. Player skill is calculated on the *Player Performance Analyzer*, by calculating a weighted average of the player's performance on all tags. Likewise, the *Player Performance Predictive System* estimates performance on the challenges based on the performance on each tag. The most suited challenge is then picked by comparing the obtained results to a *Player Performance Curve* defined by the game designer.

This is a case where player progression calculation is done resorting to an average calculation of the sliding window, from which we can conclude if the player's performance is getting higher, lower or constant.

Pardal's thesis [2] implements a similar progression model, using Holiday Knight, a bullet-hell single player video game developed by Pardal, as a testbed game instead of Smash Time, further proving this model can be correctly used when a designer looks for the player skill to follow a certain performance curve.

2.4.3.B Bicho's Progression Model

"Go, Go Hexahedron" is an endless running side-scrolling video game developed by Bicho in his thesis [20] and article [23], which can be seen in **fig.** 2.8. 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.

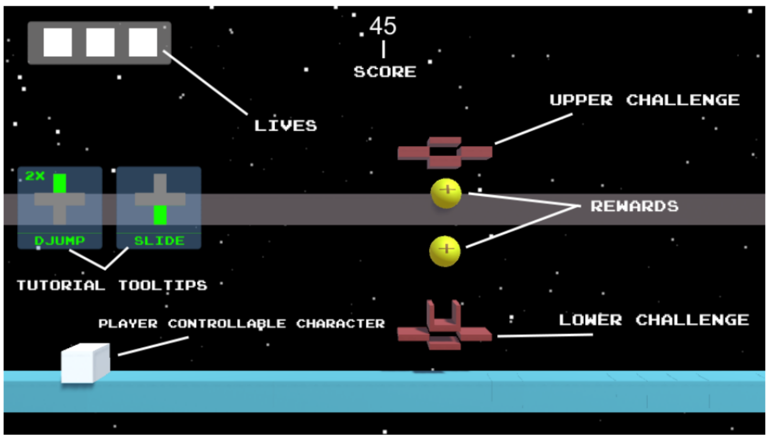


Figure 2.8: *Go, Go Hexahedron* components

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".

In this work, the player's skill S is measured in each pair is calculated through a weighted arithmetic mean:

$$S = \frac{\sum_{i=1}^{A} w_i \times A_i}{\sum_{j=1}^{A} w_j} \tag{2.1}$$

In **eq. (2.1)**, A_i represents each attempt, rated between 0 (total failure) and 1 (total success). Values between 0 and 0.5 represent negatives states, such as near-miss attempts. Values between 0.5 and 1 represent positive states. Each attempt of this "window" from the last 10 attempts contributes with a different weight w_i to the skill assessment of the gameplay element. This emphasizes that more recent attempts are more relevant than previous attempts when evaluating the player. One limitation of this model is the necessity of parameterization: the game designer needs to set values for what is a totally successful attempt, total failure and 0.5.

2.4.4 Statistical Models for Gaming Data

In this section, we will be presenting statistical models for estimating skill in video games [24], which are often used for matching players together for close matches or for qualifications for tournament purposes. Players are also interested in quantifiable measures of their own skill. New estimations can be generated after every game or after a set of games.

While there are a good deal of different gaming estimation models, the focus will now be on Elo, where we will briefly present Mestre's thesis [21] and Glicko.

2.4.4.A Elo

The Elo system started as a system to assign chess players' skill levels to adequately match them with players of similar skill. While its origins lie in chess, the Elo rating system has been expanded to other games such as board games, football, tennis and video games.

As we can see in **fig. 2.9**, upon beating their opponent, the player has their Elo updated from 1422 + 8 points. Although we can not see it, since the player earned some Elo points, their opponent must have lost a certain amount of points.

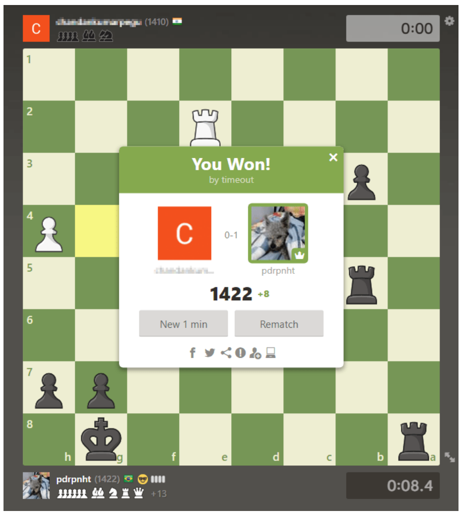


Figure 2.9: Elo system updating skill rating on Chess.com

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 [25], detecting deficiencies in fabric patterns [26] and student performance evaluations [27] [28].

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:

$$E = \frac{1}{1 + 10^{-(P-O)/a}} \tag{2.2}$$

In **eq.** (2.2), 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:

$$New = P + K(Out - E) \tag{2.3}$$

Equation (2.3) 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.

While the above shows the most basic Elo formula, there have been a variety of formulas that alter it, like altering the constant *K* to a number that is based on the number of games completed. The US chess rating system employs a K that is not constant, but instead is calculated as:

$$K = \frac{800}{N + m} \tag{2.4}$$

In **eq. (2.4)**, *N* is the total number of effective games that have been completed and *m* is the number of games in the current tournament that was played. This can be ported to the video games' scenery by keeping *N* and using *m* as the number of games played in that playing session (this value is reset upon exiting the game). "Effective" games is a measure partly depending on the number of games the player has completed or is set at a max of 50.

In his work, Mestre [21] uses the Elo ranking system to match the player with a challenge that matches exactly his skill. More than that, its progression model also predicts if his skill is progressing or regressing by measuring the player performance in previous attempts with a variable called *form*. The player has one Elo per mechanic and thus one *form* per mechanic.

The *form* of the player in each mechanic is calculated with a weighted average in a sliding window where the most recent performance have greater weight, called the *form* module. This calculation is as follows:

$$Form_m = 10 \times \sum_{i=1}^{A} i \times s \qquad Form_m \in [-550, 550]$$

$$m = Mechanic \qquad A = Sliding\ Window\ Size = 10$$

$$s = -1\ if\ failure \qquad s = 1\ if\ success$$

The difficulty of the challenge mechanic is the sum of the player Elo and *form* in that mechanic:

$$Diff_m = Elo_m + Form_m$$

In order for the challenge to match the correspondent player's skill mechanic, the dimension of the obstacle is then computing according to the previously calculated difficulty. The obstacle's dimension is calculated according to the following formula:

$$BlockDimension = MinDim + Diff_m \times (MaxDim - MinDim)$$

2.4.4.B Glicko

The Glicko formula is a variation of the Elo rating system. This formula seeks to improve estimation accuracy by having a previously fixed constant in the formula be an adjusted value. Glicko adds a standard deviation to the player's ratings in order to give information about the certainty of the rating, called Rating Deviation (RD) and is shown as RD_{PNew}, calculated as:

$$RD_{PNew} = \sqrt{\left(RD_p^2 + c^2 \times t\right)} \tag{2.5}$$

where RD_p is the current player rating, c is a constant based on how malleable RD will be over time, and t is based on how long it has been since the player has played. This formula is used with a new rating period, that is after a certain amount of time has passed between ratings, like not having played the game for a certain amount of time.

For match by match Glicko calculations, the probability of a win is very similar to the Elo formula but with a few additions:

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

where P is the player current skill rating, O is the opponent's current skill rating and a is a value to not overestimate the winning probability for high rank players. The new parameter g is added and is calculated using the opponent's RD (RD_O):

$$g = \frac{1}{\sqrt{\left(1 + 3q^2 \times \frac{RD_O^2}{\pi^2}\right)}}$$

and q is a constant, such as:

$$q = \frac{\ln 10}{400} = 0.0057565$$

The RD for the player is updated with:

$$RD_{PNew} = \frac{1}{\sqrt{\frac{1}{RD_P^2} + 1/d^2}}$$

where d^2 is:

$$d^2 = \frac{1}{q^2 \times g^2 \times E\,(1 - E)}$$

where *E* is the expected probability of winning.

Finally, all that is left is the player rating update formula which again is very similar to Elo, but with added parameters:

$$New = P + qRD_{PNew}^2 g\,(Out - E)$$

with *Out*, being the outcome for the match (1 for a win and 0 for a loss).

The Glicko distinguishes itself from Elo by adding a standard deviation to the ratings and thus allowing for the certainty of a player's rating to factor into adjustments made. A player's rating will change more if their opponent's RD is smaller, indicating more certainty that the opponent is accurately ranked. Additionally, if enough time has passed between play sessions, **eq. (2.5)** will be used to further update the RD.

2.5 Subjective Experience Measurements

Subjective measurement is how information is passed, written or verbally, by the participant while performing an experience. Besides acquiring objective data, such as number of interactions or the time to complete a certain task, it is important to "listen to our participants" and the feedback they provide. This can include using a questionnaire, where open-ended questions are asked or it is asked to rank an experience based on how they perceived it.

Although there is a wide variety of questionnaires which allow subjective experience measurement, for example, the Game Experience Questionnaire (GEQ) [29], in the following subsections we will be focusing on the questionnaires that will be used in this work.

2.5.1 Intrinsic Motivation Inventory

The Intrinsic Motivation Inventory (IMI)[26] 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.

2.5.1.A IMI Dimensions

This instrument assesses participants over six dimensions:

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

The interest/enjoyment subscale is considered the self-report measure of intrinsic motivation, thus IMI being the name of this questionnaire and this being often the subscale that has the most items.

The perceived choice and perceived competence concepts are theorized to be positive predictors of both self-report and behavioral measures of intrinsic motivation, while pressure/tension is theorized to be a negative predictor of intrinsic motivation.

Effort is a separate variable that is relevant to some motivation questions, so it is only used when relevant. The value/usefulness subscale is used in internalization studies, in other words, the idea that people internalize and become self-regulating with respect to activities that they experience as useful or themselves find valuable.

A seventh subscale has been added related to the concept of relatedness, which is used in studies having to do with interpersonal interactions, friendship formation, and so on. The validity of this subscale has yet to be established.

2.5.1.B IMI Items Selection

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.

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

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.

Items within the subscales overlap considerably, although randomizing their presentation makes this less salient to most participants. It is very important to recognize that multiple item subscales consistently outperform single items for obvious reasons, and they have better external validity. Nonetheless, shorter versions have been used and been found to be quite reliable.

2.5.1.C IMI Construction and Scoring

The first step to construct a IMI for a study is to decide which of the dimensions we want to measure, based on what theoretical questions we are addressing. We then need to select which items from those dimensions to use and randomize their order.

To use the IMI for measurement, the score of the items which were presented in the negative tense must first be reversed. This is done by subtracting the item response from 8 and use the resulting value as the item score. The next step is to calculate the core of each dimension by averaging across all the items that were presented on that subscale. Lastly, the dimension scores are used in the analysis of the relevant theoretical questions.

2.5.2 Positive and Negative Affect Schedule

The Positive and Negative Affect Schedule (PANAS)[27] 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;

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

- **Negative affect:** Hostile, Irritable, Ashamed, Guilty, Distressed, Upset, Scared, Afraid, Jittery, Nervous.

2.5.2.A PANAS Versions

Multiple versions of PANAS were created to answer various needs in subjective experimentation:

- **PANAS-C [30]:** This schedule was developed in an attempt to differentiate the affective expressions of anxiety and depression in children. The tripartite model on which this measure is based suggests that high levels of negative affect is present in those with anxiety and depression, but high levels of positive affect is not shared between the two. Previous mood scales for children have been shown to reliably capture the former relationship but not the latter. The final version of the measure consists of 27 items: 12 positive affect terms and 15 negative affect terms. Despite the purpose of its development, however, the measure's discriminant validity is still to be established;

- **PANAS-SF [31]:** This schedule comprised 10 items that were determined through the highest factor loading on the exploratory factor analysis in the original PANAS. The purpose of the PANAS-SF was not only to provide a shorter and more concise form of the PANAS, but to be able to apply the schedules to older clinical populations. Overall, it was reported that this modified model was consistent with the original;

- **I-PANAS-SF [32]:** This schedule was created separately from the PANAS-SF, in order to make a 10 item mood scale that can be implemented effectively on an international level, provide more clarity on the content of the items, reduce ambiguities, address the limitations of the original and the previous short form of the PANAS, and also to provide a shorter, yet dependable and valid scale. The positive affect subscale is formed by 5 items, as well as the negative, with the items being "Active", "Attentive", "Alert", "Determined", "Inspired" and "Hostile", "Ashamed", "Upset", "Afraid", "Nervous", respectively.;

- **PANAS-X [33]:** This schedule consists of 60 items that can be completed in 10 minutes or less. The PANAS-X incorporates the original, higher order dimensions specified in the PANAS in addition to the measures of 11 lower order emotional states. These measures are broken down into three main categories: basic negative emotion scales consisting of fear, hostility, guilt, and sadness; basic positive emotion scales consisting of joviality, self-assurance, and attentiveness; and other affective states consisting of shyness, fatigue, serenity, and surprise.

2.5.2.B PANAS Scoring

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.

2.6 Summary

In **chapter** 2 we started by studying the concept of flow and its effect on the player: an extreme concentration on the task being performed with the loss of self-awareness and distortion of the experience of time. This is the effect we want to keep our players in when playing a game of our design.

One of the rules to keep in mind in order to keep the player in this flow state is demanding actions to achieve goals that fit within their capabilities or, in other words, have a player successfully complete a challenge composed by certain game content and with a certain level of difficulty associated. This is not such a straightforward task, specifically when the game scenarios are different in each run, we are uncertain on what difficulty to play and are unable to calculate the player's skill level.

PCG aims to create more game content variety, thus reducing the level of boredom and increasing replayability. There is a wide variety of PCG types, as presented in **section** 2.2.2, with multiple dimensions solutions and each with their pros and cons. Despite all these techniques that we have in our dispose, the biggest problem we are to face is ensuring the generated game content has an associated level of difficulty that is suitable to the player's skill.

When discussing difficulty in video games, the first idea that comes to our mind is the multiple difficulty options that are presented upon starting the game. Static difficulty, as presented in **section** 2.3.1, provides an environment where the player must obtain a certain level of skill to surpass specific challenges, which may become frustrating if the player is unable to perform the right mechanic or solve the intended puzzle and end up with the player quitting the game. Dynamic Difficulty Adjustment allows the game to adapt to the player's skill by providing challenges with adequate difficulty. If done correctly, this technique allows for every player to find the game challenging, independently of their skill level, increasing the player's interest and engagement and thus getting them in the flow state.

In order to know the level of difficulty of the challenge we want to provide the player, we must be able to calculate and define their skill level. In **section** 2.4 we start by looking what characterizes both a player model and a skill atom and the importance of defining a skill and progression model. Catarino [19] and Pardal [2] used a skill-based progression model that used tags for the game designer to associate with each challenge and then rate the player's performance when facing it. Bicho [20] developed a model where performance over time was taken into account and allowed the player to choose which dimension of challenges to face. Mestre [21] introduced the Elo rating system in a single player game and used it in order to calculate player skill. Furthermore, Mestre also used a form module to take into account performance over time. All the before mentioned work show the importance of having a model which is able to calculate and properly define a player's skill level according to their performance, in order to find the most suitable challenge for the player.

Lastly, we want to be able to measure both objective and subjective experience information. While objective measurements can be easily obtained by having its collection prepared, subjective data is related to each player, thus being a harder type of information to collect and analyze. In **section** 2.5 we can see how to gather and measure data related to each participant's perceived experience in a multitude of dimensions, as well as a score for their positive and negative affectivity level.

3
Testbed Game

Contents

3.1 Choice of the Testbed Game

In order to test the formulated hypothesis, the testbed game would have to be simple enough so that all types of players could easily start playing, but with a skill gap wide enough so that experienced players could come up with strategies which would allow them to face harder challenges and reach higher scores and engaging enough so that everyone would have fun playing the game and staying on the flow state.

The game that was chosen that was able to fulfill all these needs was Holiday Knight, a bullet-hell single player video game developed by Pardal in his Master's thesis.

3.2 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.

Multiple changes had to be made to Pardal's developed version of Holiday Knight in order to test the formulated hypothesis. The player movement was kept and actions such as shooting were kept. All available character actions will be further discussed in the next section.

3.3 Character Actions

From the moment the game starts, the player enters the first room and is presented what inputs, which can be seen in **fig.** 3.1, will reflect on character actions such as:

- **Moving**: When pressing the W, A, S, D keys, the player will move up, left, down and right, respectively. This is the base action which will allow the player to move their character through the rooms and to dodge enemy bullets;

- **Aiming**: By moving the mouse, the player will be able to aim. When taking into account the mouse position, the character will look and point their weapon in that direction;

- **Shooting**: By pressing the left mouse button, the player will be able to shoot in the direction they are aiming. Each of these bullets that hit an enemy will remove 1 health point of their total health pool. Single presses of this button allow for a more precise shooting and slower fire rate, while keeping the button pressed will provide the fastest fire rate of 5 shots per second. This shooting action is only possible when the weapon has at least 1 of the maximum 20 available bullets and is not reloading;

- **Reloading**: By pressing the right mouse button, the player will immediately reload their weapon, thus interrupting any shooting action. Upon starting the reload animation, it will take a defined time of 2.6 seconds for the reload to finish and getting back the defined value of 20 available bullets, which can then be shot. When the player shoots the last available bullet, the reload animation automatically starts, thus preventing the player from trying to shoot when there are no available bullets;
- **Throwing weapon**: When holding a weapon, by pressing the Space key, the player will always be able to throw the character's weapon. When the weapon is thrown, it can hit multiple enemies until it eventually stops. Each hit from a thrown weapon will subtract 7 health points from the enemy's total health pool. This action can be done even when the player is reloading, which in this case will stop the reload action. Upon picking the weapon back by colliding with it, the player is now free to shoot with it, reload or even throwing it again.

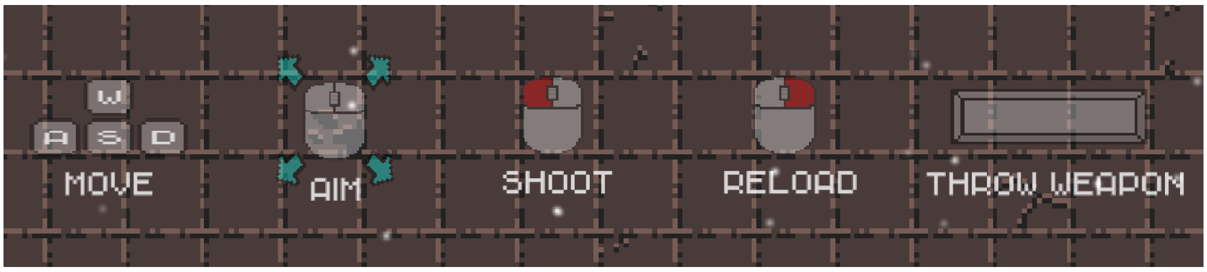


Figure 3.1: Game controls depicted in the first room

3.4 Game Loop

When the game starts, the player find themselves on the first room, where they are introduced to the game controls and where they must pick up their weapon. Upon shooting the interactable target on the corner of the room, they are now able to move on to the next room.

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 player loses 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. The formula used for the points system, where *P* represents the amount of points earned in the current room and *t* the number of seconds spent in the room from the moment it was entered to being cleared, can be seen below in **eq. (3.1)**:

$$P = \begin{cases} \max(0, (60 - t) \times 1000), & \text{if player clears the room} \\ 0, & \text{if player dies} \end{cases} \tag{3.1}$$

With this formula we can see that the theoretical max amount of points the player can earn in a room is 6000 if they were able to clear it as soon as they enter. For each second that passes, the player will earn less 100 points. We can also verify since there are a total of 20 rooms in a run, the maximum points the player can earn, theoretically, is 120000.

3.5 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.

3.5.1 Rooms

In every run of the game, the player will always find themselves on a similar initial room, which is depicted in **fig.** 3.2.

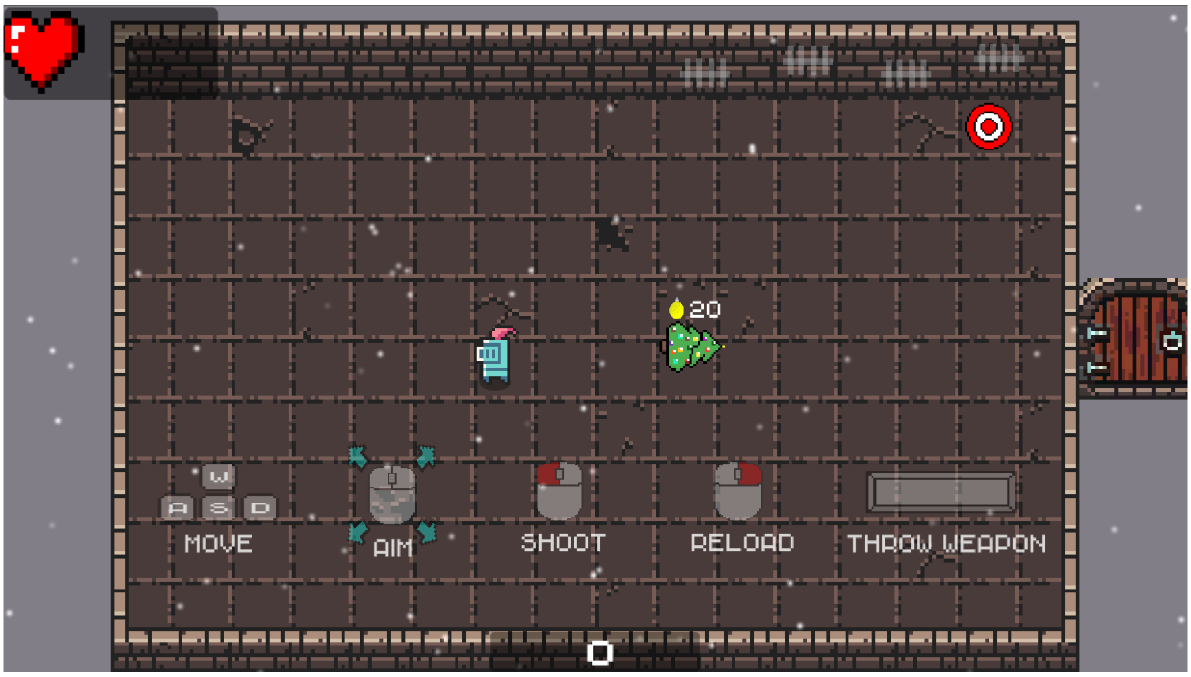


Figure 3.2: Initial room of Holiday Knight

In this picture we can see the player close to the center, represented by the blue knight close to the center, the weapon which can be picked up, represented by the green Christmas tree with the maximum number of bullets above it, in this case, 20.

We can also find the already described controls overlay, the points overlay, which is initiated at 0 and updated after the completion of each room that has an enemy team, and the health overlay which will show the player's health once they enter the first room with enemies and is updated every time the player is hit and reset after each room's completion.

A brown door can be seen on the right of the room, which will only open after each room's completion and close after the player moves to the next room. In the case of the initial room, the red and white target in the top right corner must be hit for the door to open.

Finally, every room has 20 white chalk marks in the top corner. For each of the completed rooms, a chalk mark will turn yellow, allowing the player to keep track how many rooms are left until the end of the run.

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. An example of this last room can be seen in **fig.** 3.3.

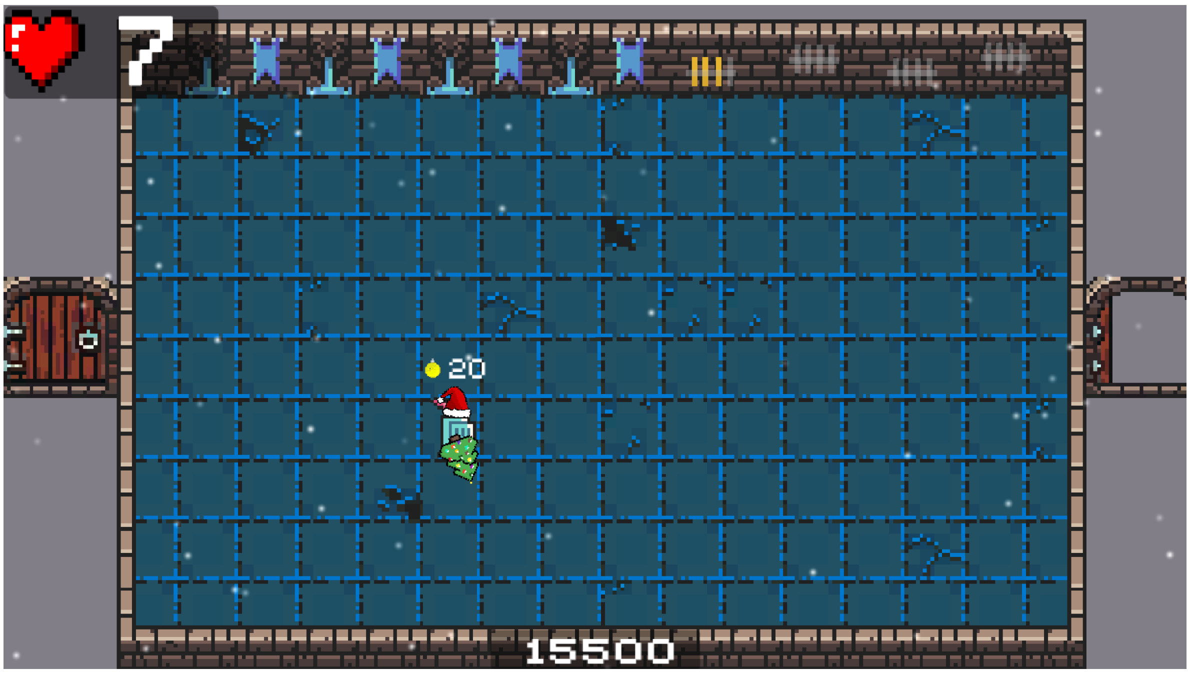


Figure 3.3: Example of a water room environment

The effects of each room environment are discussed on 3.5.3.

3.5.2 Enemies

There are a total of 10 different enemies, which can be seen in **fig.** 3.4, named: Big Zombie, Chort, Ice Zombie, Imp, Masked Orc, Muddy, Orc, Shaman, Skelet and Swampy.

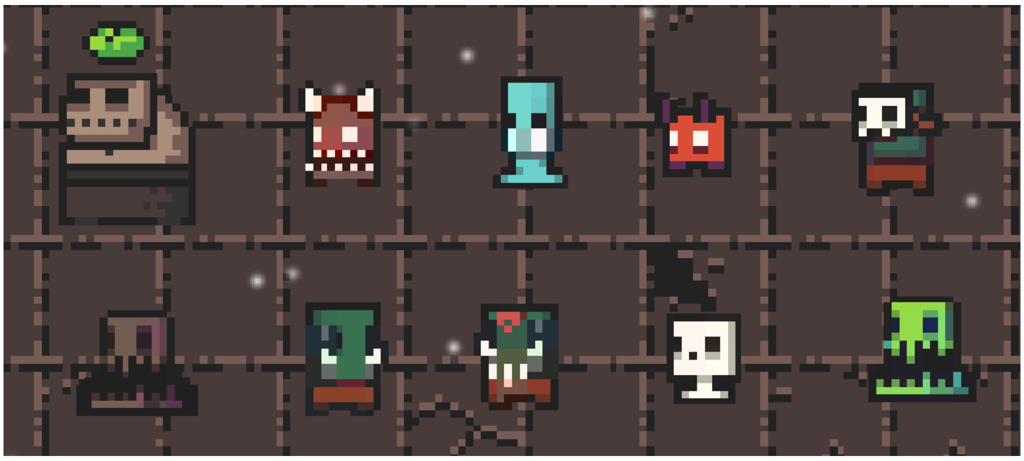


Figure 3.4: The 10 different enemies present in the game

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 and damage per shot.

More information regarding the enemies and their behavior can be found on Pardal's master thesis.

Lastly, each enemy also has a type, which is defined by the color of its aura. Just like the rooms, the red aura means the enemy is a fire type, blue aura means water type and green aura means grass type. According to the enemy type and the current room environment advantage or disadvantage, these last three referred parameters can either be boosted or reduced, as explained in 4.1.1. An example of an enemy and its visible aura can be seen in **fig.** 3.5.

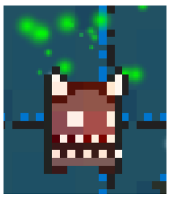


Figure 3.5: A grass Chort in a water room

3.5.3 Enemy Type and Room Environment Synergy

As previously mentioned in 3.5.1, there are a total of 4 possible room environments: neutral, fire, water and grass. Similarly, as referred in 3.5.2, players may find enemies with 3 types, which are the same as the room, except for the neutral one.

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 graph in **fig.** 3.6.

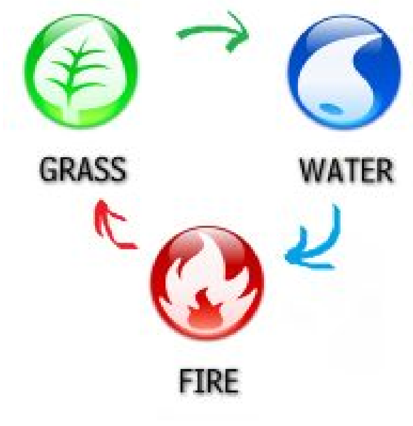


Figure 3.6: Graph of the types trio

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.

4 Implementation

Contents

The developed versions of this work were implemented in Unity[1]. Unity is a cross-platform game engine developed by Unity Technologies. This game engine supports a variety of desktop, mobile, console and virtual reality platforms with a high content variety regarding techniques and content that one is able to produce with the engine. Unity offers a free subscription to indie game developers and is one of the most used public engines in the gaming market, aiding with the creation of games like Pokémon Go[2], Monument Valley[3] and Cuphead[4].

The Unity build version that was used in this project is Unity Version: 2020.3.21f1, since this is the earliest version from which we are able to upgrade the original Holiday Knight project folder at the time of starting writing (October 2021). Version 2020.3 is also a Unity LTS version, which means it provides a stable foundation for projects that are in production. This version is specific has biweekly updates until mid-2022, after which updates will be monthly until March 2023[5].

The programming language that's used in Unity and that was used in the project is C#, which is an object-oriented programming language. C# scripts will be developed in the most updated version of Visual Studio[6], which is an integrated development environment for the Windows operating system, packing an array of tools and features to elevate and enhance every stage of software development.

In this work, we hypothesize that the player prefers a version of a PCG single player video game where the challenge's difficulty is adapted to their performance (A version), which is measured with an Elo-based statistical model, over a version where challenges are ordered according to their difficulty (NA version). The player should also prefer this last version over one where the challenges are randomly selected (R version).

In order to test this, 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.

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

[1]https://unity.com/
[2]Pokémon Go (Android, iOS, 2016) is an augmented-reality mobile video game developed and published by Niantic.
[3]Monument Valley (Android, Windows, 2014) is an indie puzzle video game developed and published by Ustwo Games.
[4]Cuphead (Windows, Xbox One, 2017; macOS, 2018; Nintendo Switch, 2019; PlayStation 4, 2020) is a run-and-gun developed and published by Studio MDHR.
[5]https://unity3d.com/pt/unity/qa/lts-releases
[6]https://visualstudio.microsoft.com/

implemented.

4.1 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.** 3.4. With this enemy variety and enemy teams' composition, a total of 5850 challenges can be generated, as can be seen in **eq. (**4.1**)**:

$$\mathrm{A}_2^{10} + \mathrm{A}_3^{10} + \mathrm{A}_4^{10} = 5850 \tag{4.1}$$

Although this appears to be many generated challenges, the chance that a player faces the same challenge twice is not that rare. In order for this chance to be around 50%, a total of 90 challenges must be generated. This calculus is done following the *Birthday Problem* [7], which expression can be seen below, in **eq. (**4.2**)**, with $P(n)$ = 0.5:

$$P(n) = 1 - \left(\frac{5850!}{5850^n \, (5850 - n)!} \right) \tag{4.2}$$

This means that on the sixth run (taking into account there are 20 challenges in each run), the player has a probability of 50% to find repeated challenges, which will eventually break immersion and drive the player away from the flow state.

With the dynamic challenge adaptations that are presented next, 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.

4.1.1 Dynamic Enemy Parameters

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 to see the situations where an enemy has an advantage or disadvantage over the environment type, refer to 3.5.3.

On **table** 4.1 we can observe the changes on the enemies' base parameters according to their possible levels.

[7] https://en.wikipedia.org/wiki/Birthday_problem

Levels/Parameters	Damage	Health	Speed
Reduced	1 hitpoint	-6 health points	$\times$0.4
Low	2 hitpoints	-3 health points	$\times$0.8
Normal	3 hitpoints	+0 health points	No change
High	4 hitpoints	+3 health points	$\times$2
Boosted	5 hitpoints	+6 health points	$\times$3.5

Table 4.1: Enemy parameters change according to their level

In order for these changes on the enemies to be noticed by the player, visual differences were added. In **fig.** 4.1 we see the five types of bullets the enemies can shoot. The higher the level of its damage parameter, the darker the bullet is. This way, the player should prioritize avoiding damage from the strongest hitting enemies in order to preserve their own health points.

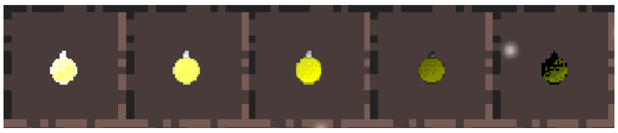


Figure 4.1: Bullets shot by the enemies according to their damage parameter level

Likewise, in **fig.** 4.2 we see the five different colored health bars according to their health parameter. The closer to the purple hue, the more health the enemy will have. This way, the player should take into account which enemies have to be targeted longer to be defeated.

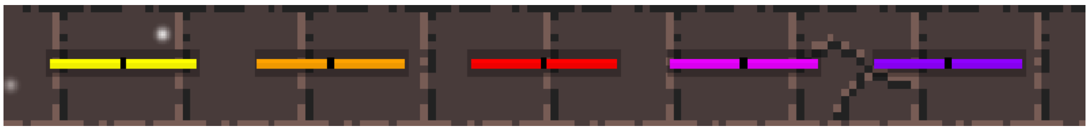


Figure 4.2: Enemies health bars according to their health parameter level

The speed parameter is immediately visible by the player when they observe each enemy move. For this reason, no other visible cues were added to the enemies, which also avoids the addition of possible visual noise on each challenge. The speed of each enemy changes according to its parameter level, by multiplying its base speed by a defined value, as depicted in the last column of **table** 4.1.

4.2 Challenge Generation

As will be further approached in chapter 5, we have to ensure every player who tests each of the three developed versions has the same pool of generated challenges, in order to avoid having a player face stronger enemy teams which could hinder the way they experience the game.

To resolve this issue, we created a *.csv* file, 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. The A version, which is explained in section 4.9 would specifically be affected by this. 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:** Although in this work, the picked number was 1000, this can be changed if more or less enemy teams are wanted;
- **Selecting number of enemies in each challenge:** This does not need to be a concrete number, but a range. In this work, each challenge can be formed by 2 to 4 enemies. The number of challenges with the same number of enemies grouping in a team should be close, if not, this step is repeated until it happens.
- **Selecting enemies for each enemy team:** From the pool of 10 different enemies, we randomly assign them for each enemy team, taking into account its defined number of enemies. Each enemy can be assigned more than once. For example, one challenge can be formed by 3 Chorts;
- **Selecting the base level for each of the enemies' parameters:** All enemies' damage, health and speed are assigned a *Low*, *Normal* or *High* base level. Levels such as *Reduced* and *Boosted* can only be dynamically assigned according to the enemy's synergy with the room environment;
- **Selecting the enemy type for each enemy:** Each enemy is assigned one of the three types: fire, grass or water.

4.3 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, which can be seen in **table** 4.2, and the percentage of victories for each of the five levels of each dynamic parameter, was collected on two *.csv* files.

Enemies	**Big Zombie**	**Chort**	**Imp**	**Masked Orc**	**Muddy**	**Orc**	**Shaman**	**Skelet**	**Swampy**	**IceZombie**
Big Zombie	53%	100%	72%	88%	74%	56%	54%	97%	61%	96%
Chort		48%	7%	10%	12%	9%	2%	43%	0%	26%
Imp			53%	97%	100%	57%	39%	99%	57%	77%
Masked Orc				50%	97%	34%	0%	0%	31%	90%
Muddy					50%	2%	1%	0%	2%	60%
Orc						56%	54%	84%	91%	76%
Shaman							54%	89%	60%	97%
Skelet								50%	5%	55%
Swampy									52%	93%
Ice Zombie										54%

Table 4.2: Percentage of victory of each enemy

In this table, we can observe the Big Zombie won 72% of the one thousand fights over the Imp. Looking at the diagonal of the table, we can also see that when an enemy fights its similar, the resulting percentage is very close to 50%, which is what would be expected in this scenery.

4.4 Challenge Selection Heuristic

Simulating every needed fight between enemy teams would result in big waiting time intervals, hindering the player experience. In order 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.

There are multiple ways an issue like this can be tackled, like using Artifical Neural Networks (ANN), which are computing systems inspired by the biological neural networks that constitute animal brains. In this case, a heuristic should be enough to define which enemy teams are the strongest and consequent challenges with a higher difficulty level, since we are using a few discrete variants that correspond to the number of enemies in the enemy team and each of their parameters. Since we also know the rules which support the game mechanics, from being the game designers, we can also take advantage of this knowledge to formulate the heuristic.

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. Let enemy team X be formed by enemies A, B and C and enemy team Y be formed by enemies D and E. In order to calculate the heuristic for each team, we start by calculating the heuristic value of each enemy, such as H_X being the value for enemy team X and H_A the value of enemy A, as follows in **eq. (4.3)**:

$$\begin{cases} H_X = H_A + H_B + H_C \\ H_Y = H_D + H_E \end{cases} \tag{4.3}$$

The heuristic value of each enemy is calculated using, in the case of enemy A, P_A and V_A, representing the percentage of wins of each dynamic parameter of A and the percentage of wins of A over each of the enemies in the other team, in this case, enemy team Y, respectively. The heuristic formula continues in **eq. (4.4)**, **eq. (4.5)** and **eq. (4.6)**:

$$H_A = P_A \times V_A \tag{4.4}$$

$$P_A = Damage_A + Speed_A + Health_A \tag{4.5}$$

$$V_A = \frac{V_{AD} + V_{AE}}{N_Y} \tag{4.6}$$

In this last equation, **eq. (4.6)**, we observe V_{AD}, which represents the percentage of wins of enemy A over, in this case, enemy D and N_Y the number of enemies present in the enemy team that A is facing, in this case, Y.

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, it is now possible to order each of the challenges according to their number of wins, which represent how strong each enemy team is and hence its difficulty.

4.5 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 4.3**.

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. This means, it is possible for enemy team X to win three times over enemy team Y and lose the other two 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, since it does not simulate perfectly how the fights would turn out. But this heuristic and its simplicity is also advantageous: it performs a high number of simulations in a short amount of time and with minimal resources and, opposite to structures as ANNs which would have to be reset, in the case of new content is added, the heuristic is adapted by the designer.

4.6 Developed Models PCG Properties and Taxonomy

Before looking into each of the developed models, it is important to evaluate their PCG properties and taxonomy, so it is clear where this generation of content is integrated.

In the developed versions of Holiday Knight, content generation occurs in the generation and selection of challenges for each of the rooms the player goes through. This process has to be fast enough, so the player does not stay waiting for each challenge selection. It also needs to be reliable, so there are no challenges created with invalid criteria, and controllable enough so that the designer may be able to change the parameters of each enemy and its team.

In all developed versions the enemy teams are generated prior from the player facing them, with this generation being necessary for level completion, but enemy team selection in version A is made on runtime (online). All these enemy teams are automatically generated, according to parameters made available by the designer, in one pass. R and NA versions are generic, opposing the A version, which is adaptive, since it takes into account the player performance in order to select the next enemy team the player will be facing.

4.7 Random Challenge Model

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

For testing purposes, as will be explained in **chapter** 5, a way for all the play-testers to have the same randomly generated enemy teams selects for the 20 rooms had to be ensured. A checkmark in the Unity scene where the random challenge model was implemented was added so the generation and selection of the challenges could quickly be switched between totally randomness and pre-selected random enemy teams from the *.csv* file where the 1000 pre-generated and ordered by difficulty challenges had already been generated.

The selected challenges for this pseudorandom selection are as follows in **fig.** 4.3:

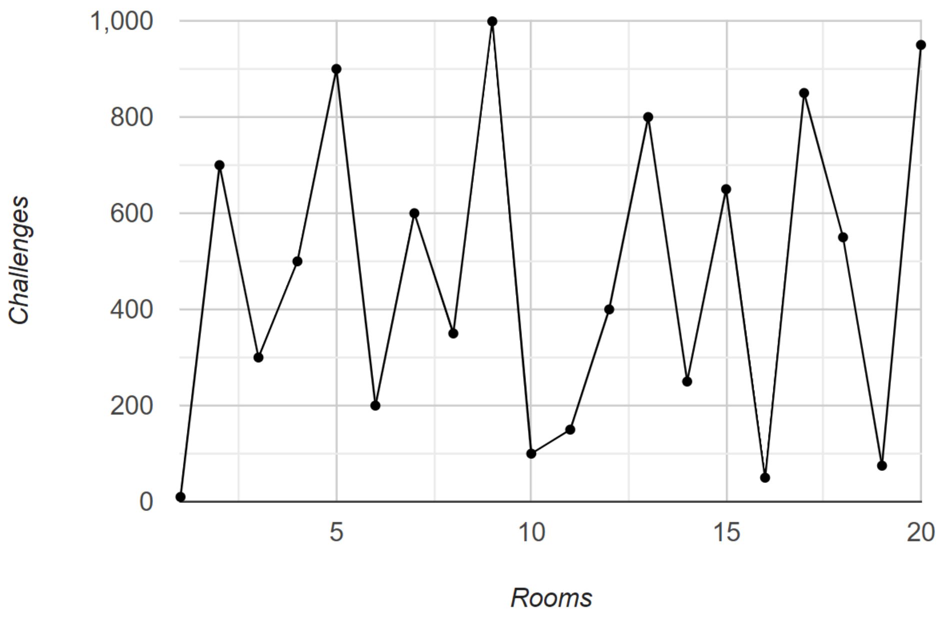


Figure 4.3: Pseudorandom selection for random challenge model testing

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. Two specific cases where we can clearly observe this pseudorandom challenge selection is from room 9 to room 10, where the difficulty drops from maximum to almost minimum, and room 19 to room 20, where difficulty increases from almost minimum close to maximum.

4.8 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.

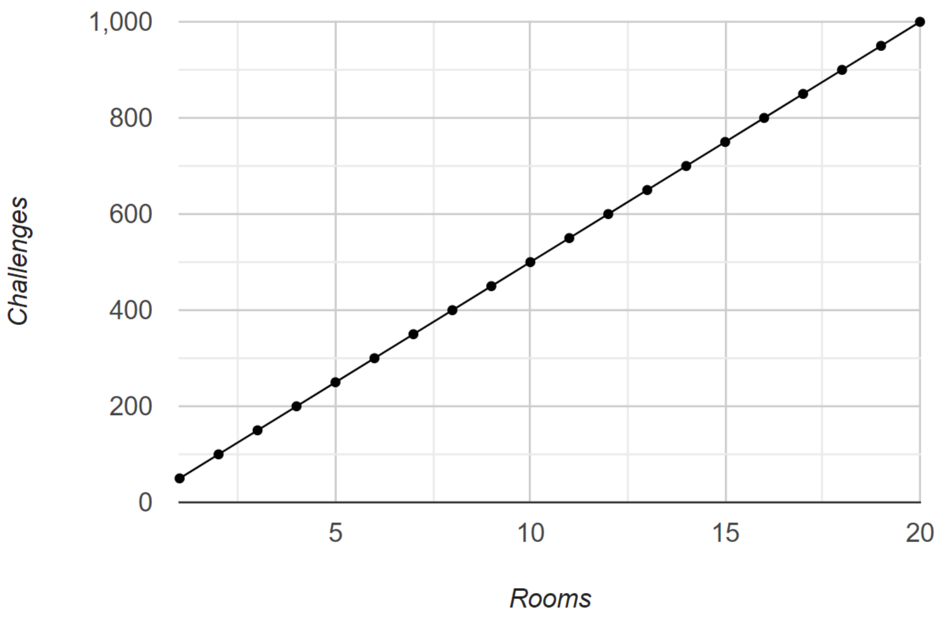


Figure 4.4: Selected challenges with increasing level of difficulty

In **fig.** 4.4 we see 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. In this case, if we want to retrieve the challenge that corresponds to a certain room, all we have to do is attribute the number of the room we want to generate the challenge for to x and retrieve the y line of the *.csv* file.

4.9 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.

Both the player and each enemy team start with a set Elo value representing their skill level. In this work, the value used was 1000, which will allow to obtain values in the range of 0 to around 3000, but can be easily changed if different ranges of Elo values are desired.

Upon a 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. With a value of 20, a total of 19980 fights are predicted in the smallest time possible, keeping this model load time close to the other two models previously addressed, thus not keeping the player waiting for the fights to occur and affecting their experience.

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, totalizing 499 fights to be simulated by the heuristic;
- Upon fight completion, the Elo rating of each team (player and all enemy teams) is adjusted according to if they win or lose;
- 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.

In the case where more than one run of this model is played consecutively, there is no need for other bootstraps. Upon one run ending, the Elo rating of all enemy teams and the player are registered and are loaded for the following runs.

4.9.1 Elo Statistical Model Implementation

Although two possibilities of statistical models for gaming data were presented, since we are not taking into account the time variable in our performance measurement, which would be the main advantage of using Glicko, we chose to implement the Elo system for our skill rating calculations. Furthermore, since we will be having a high number of enemy teams fighting each other as the player is facing their challenge and, in order to not affect the player's experience, we chose the statistical model that would be less algorithm-intensive from the ones we presented.

The formulas used to calculate the updates on the Elo rating on both the player and the enemy teams are the ones already specified on **section** 2.4.4.A.

The used formula for the expected probability of winning is the one presented in **eq. (**2.2**)**. This formula is used for both player versus enemy teams and enemy teams vs enemy teams, with *E* and *P* representing each of the entities that are fighting.

The updated Elo rating is calculated using **eq. (**2.3**)** with *E* and *P* representing each of the entities that are fighting, *Out* be the outcome of the fight, with 1 for a win and 0 for a loss.

4.10 Tutorial

The last thing to be implemented was the tutorial. This should introduce the player to our implemented version of Holiday Knight and its many mechanics and features.

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, like shown in **fig.** 4.5:

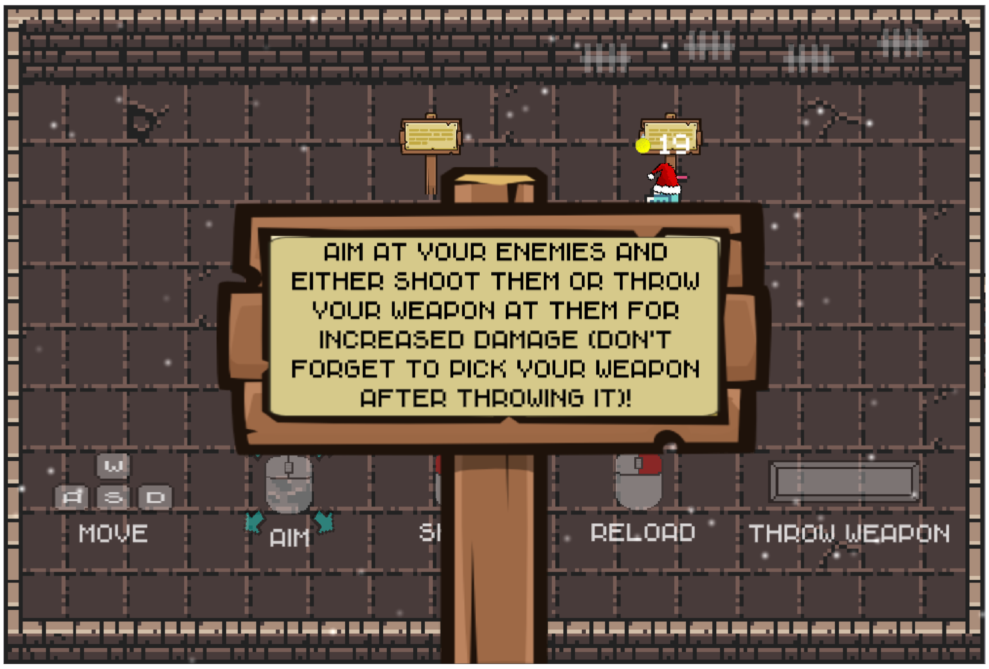

AIM AT YOUR ENEMIES AND EITHER SHOOT THEM OR THROW YOUR WEAPON AT THEM FOR INCREASED DAMAGE (DON'T FORGET TO PICK YOUR WEAPON AFTER THROWING IT)!
W
A S D
MOVE
AIM
SI
RELOAD
THROW WEAPON

Figure 4.5: Tutorial sign introducing aiming, shooting and throwing weapon mechanics

The composition of the tutorial rooms is as follows:

- **First room:** The 10 enemies are present and it's indicated that they have different shooting patterns, behavior and base damage, health and speed. It is also explained the types the enemies can have and which have advantages or disadvantages over which;

- **Second room:** The different visual differences between the levels of the dynamic parameters of the enemies are shown. Although the speed has no visual indicator besides the own speed of the enemy, there is also a sign indicating enemies may be faster or slower;
- **Third room:** The goal of the game is presented, which is completing the 20 rooms, each as fast as possible in order to earn more points. There is also a warning indicating the next room will have the first enemies the player will be facing;
- **Fourth room:** An enemy team composed by a Chort and a Swampy is generated. Both these enemies have weak parameter levels. Only after the challenge is completed, a sign is spawned indicating that during the game the player will only have 7 lives, which will be restored after clearing each room;
- **Fifth room:** An enemy team composed by an Imp, Masked Orc and Muddy is spawned. Although there are more enemies, only the Muddy has normal parameters, with both the Imp and the Masked Orc having weaker parameter levels. After the challenge is completed, a sign is spawned explaining each room may have 2, 3 or 4 enemies and indicating the next room will be the tutorial's final challenge;
- **Sixth room:** An enemy team composed by an Ice Zombie, Big Zombie, Skelet and Orc are spawned. All these enemies have stronger parameters. After defeating the whole enemy team, the tutorial ending animation is played and the player returns to the main menu.

While going through the rooms where enemies spawn, 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.

After completing the tutorial, the player is ready to play our developed version of Holiday Knight.

5 Evaluation

Contents

In this section, we will be presenting the evaluation phases of this work, which should be able to provide an answer to the initial prompted hypothesis of what version of a PCG single player video game the player prefers.

The evaluation started with a preliminary evaluation, followed by the final evaluation and the statistical analysis of the results that were obtained.

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, as depicted on section 2.5.2.A.

A questionnaire was prepared using Google Forms[1] where the initial page of the questionnaire presents our work and ensures the player is informed regarding their testing participation being completely voluntary and that they are free to interrupt it at any point in time. The questionnaire's next section gathers players' demographic information, followed by three similar sections, in which for each version, the aforementioned subjective information was measured and how the player felt the challenges were generated. The questionnaire finishes by asking the most and least preferred version and the reasons behind this decision, followed by thanking the player for participating in the testing of our work.

5.1 Pilot Evaluation

The goal of the preliminary evaluation is to prepare grounds for the final evaluation, so all the planned procedures are clear and allow for the whole experiment to go as smoothly as possible.

There were multiple reasons to perform the preliminary evaluation, such as:

- Confirm the concept we developed based on Holiday Knight is fun and conveys a good experience to the player, which should be able to understand the game's objective to clear the 20 rooms in each run in the lowest possible time;
- The created tutorial should be clear enough for the player to understand what challenges they will face and what mechanics can be used for their advantage;
- Validate the heuristic implemented for the combat simulator, since the players will see each challenge presented by the enemies differently from the game designer;
- Get feedback from the game mechanics and identify bugs in the game logic and models used to measure skill. This will allow integrating possible feedback and deal with bugs that were not

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

noticed during development;

- Verify if the questions from the questionnaire are clear and if all the information we intend to gather is obtainable.

5.1.1 Procedure

In the preliminary evaluation, each player started by playing through the tutorial. Upon completing it, players were asked if information was clear or if something specific might have been too confusing. They were also asked how they felt regarding the difficulty of each of the three enemy teams they faced in the tutorial rooms.

Upon annotation of the players' answers regarding the tutorial and answering the questionnaire's demographic questions, they played each of the three developed versions for a total of two times.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.

After playing each version, the players answered the questionnaire corresponding to the version they had just played and, after playing all of them, each player answered the last section of the questionnaire, regarding their version preference.

Lastly, each player was asked regarding their general experience while playing, any weird moment in the game that might have been originated from a bug, how they felt throughout the testing and if there was something they would like to see changed.

5.1.2 Results and Changes

A total of 6 people participated in the preliminary evaluation. Although most of them thought there was a lot of information being conveyed in the tutorial, all understood the concepts of the game, reporting that, before experimentation, memorizing which colors corresponded to the enemies that had their health points increased or decreased the most seemed to be the hardest mechanic. All players also reported that they felt the three enemy teams that were introduced in the tutorial were of increasing difficulty, as we were expecting.

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.

Quoting some players regarding the level of difficulty of the challenges that were presented in each version:

- **Version X:** “I felt really confused, as difficulty did not seem to follow any sort of logic” and “Difficulty seemed to spike and drop randomly”;
- **Version Y:** “Although some rooms felt easier than some I had previously cleared, difficulty seemed to generally go up”;
- **Version Z:** “It seemed too easy in the beginning, but when I was about half of the first run, it became a lot more challenging” and “At first it was going really well, but by the end it was just too tough”.

All these quotes were accurate in defining the implemented model that was being tested. Version X does not follow logic and spikes and drops, since its difficulty is random; Version Y’s difficulty does go up linearly, but since it was defined according to a heuristic, one player might find one challenge harder, while the other player finds it easier, which explains why some challenges might be interpreted as easier than some that appeared before; Version Z’s quotes look the opposite, but they both represent the model quite well, since an experienced player might find the first challenges too easy and only find them to be challenging enough upon reaching a higher Elo rating, while a less experienced player might find it easy in the beginning and manage to clear some rooms, but upon reaching the ceiling of their skill, they start to lose to more challenging enemy teams.

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).

Players also claimed that although the testing took a lot of time (around 60 minutes), they did not feel bored and, on the contrary, felt their experience while playing to be quite engaged and the game was fun to play.

Lastly, players were able to identify two minor bugs that would need to be fixed before the final evaluation.

5.2 Final Evaluation

The goal of the final evaluation is to obtain relevant data which will allow us to provide a sustained answer to the stated hypothesis. This information is gathered using the participant’s feedback and the collected data from both the play session and the questionnaire, which is annexed on appendix A. Questions 9 to

16 were removed since these were duplicates of the sections that are prompted related to one version and that are similar to the other two versions.

In order to obtain the needed information, we intended to gather a minimum of 20 participants that would play all three developed versions of Holiday Knight after completing the tutorial and answer the full questionnaire.

5.2.1 Procedure

The procedure of the final evaluation closely resembles the procedure of the preliminary evaluation depicted on section 5.1.1.

The participant is in the presence of an overseer, whose objective is to annotate the participant's commentaries throughout the whole play session and to be sure each task is presented in the correct order. The overseer does not interact with the participant in any way.

The participant starts by reading the first page of the questionnaire, where they give consent in being informed their participation is voluntary and they are free to interrupt it at any point. The participant advances then to the next page, where questions related to the participant's demography and gaming habits are asked. By the end of the page, the overseer selects to what section to move on. This selection is based on the order of the versions the participant is attributed at the beginning. This attribution is random, while keeping the number of tests for each of the 6 possible orders the same.

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.

In case the whole procedure goes smoothly, it should not take more than 45 minutes to complete the user test.

5.2.2 Results and Changes

A total of 30 participants took part on the final evaluation procedure. Their ages vary from 20 to 27 years old, as can be seen in **fig. 5.1**. 87% of the participants were male, while the remaining 13% were female.

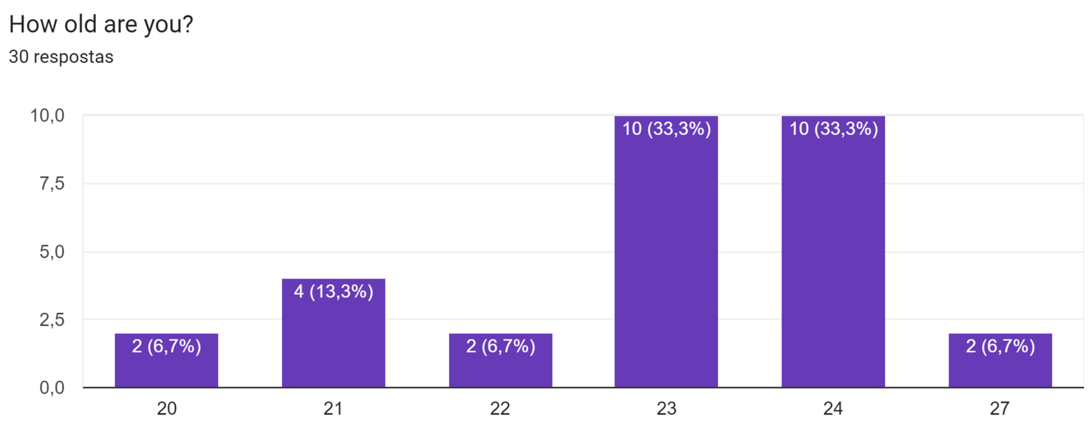


Figure 5.1: Age of the final evaluation participants

19 of the participants answered they make time on their schedule to play video games, while 10 reported they play occasionally when the opportunity presents itself. Only 1 participant answered they did not 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, the majority of the participants. more specifically, 22 of them, answered they enjoy these games and have played/watched others play them multiple times, as depicted in **fig.** 5.2.

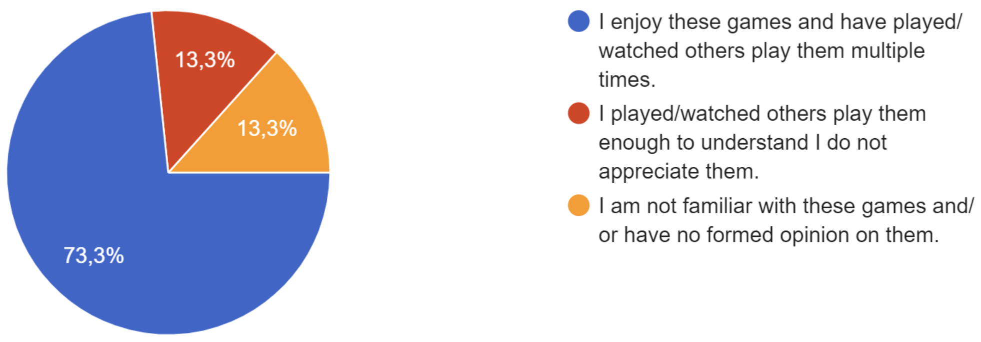


Figure 5.2: Familiarity of the participants with Holiday Knight-like video games

With each participant playing each of the three versions twice, the score from a total of six runs was recorded. The minimum, maximum and mean scores were registered, as well as the value for the standard deviation, as shown in **fig.** 5.3.

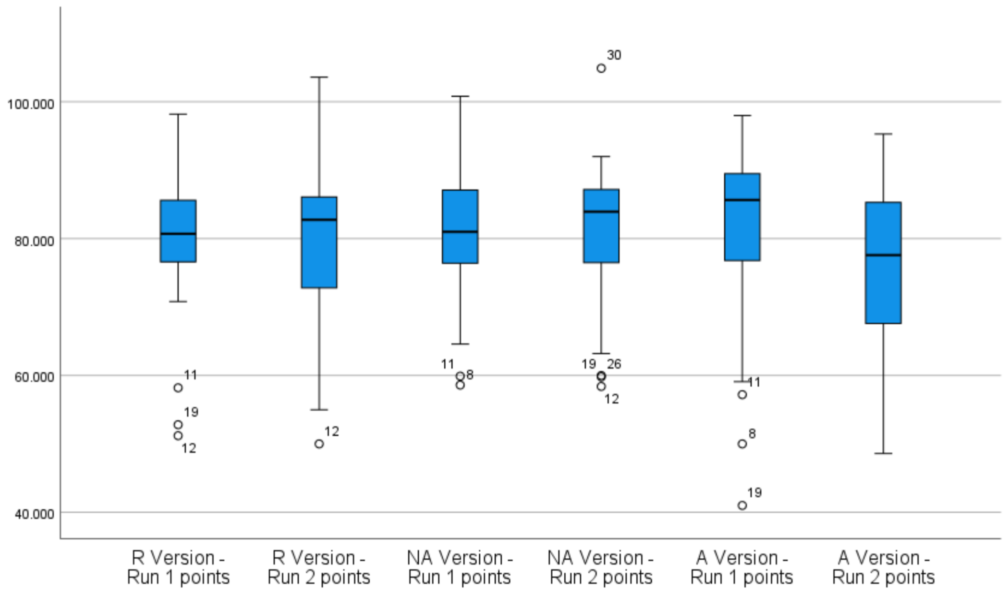
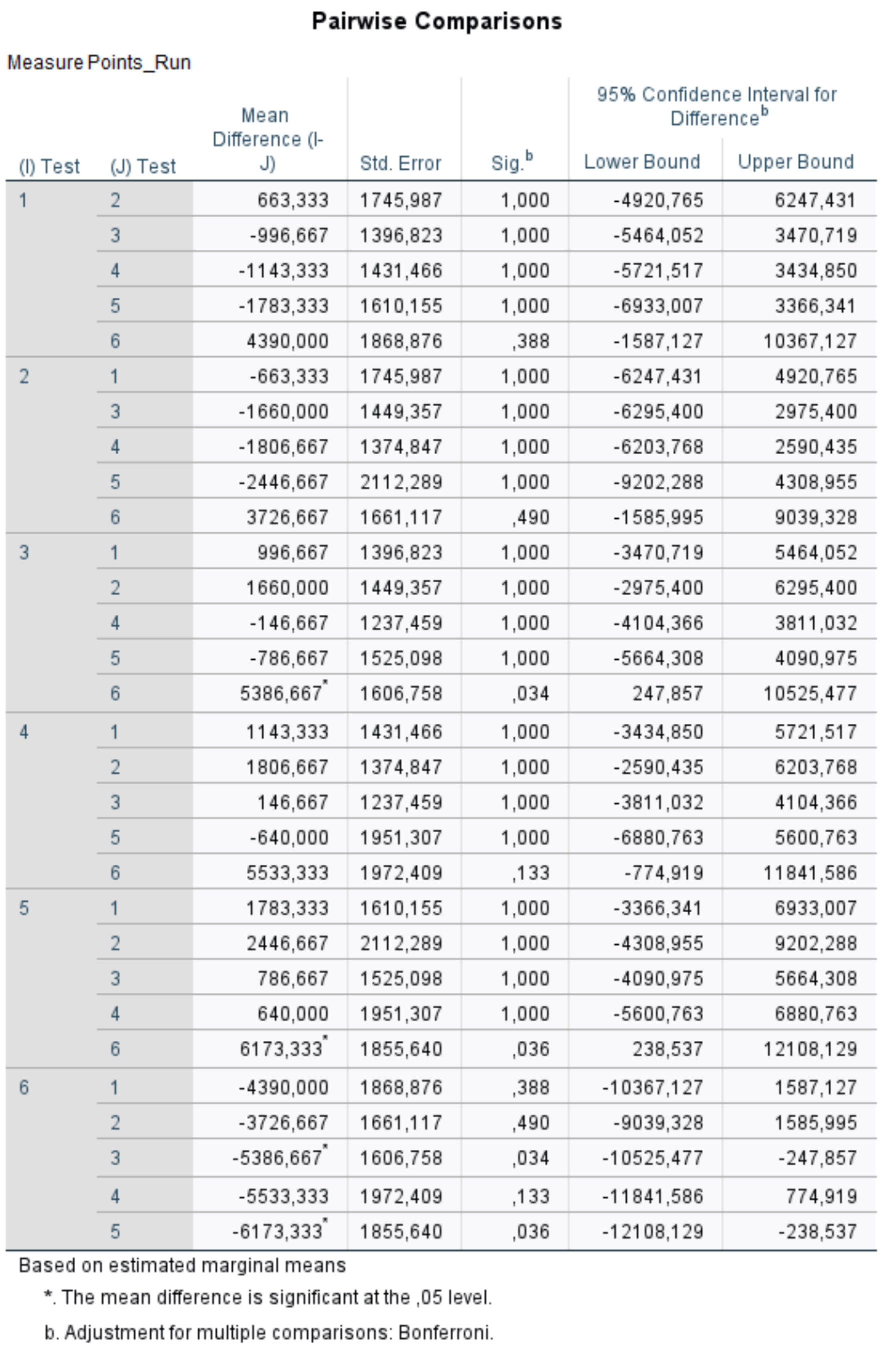


Figure 5.3: *Repeated Measures One-Way ANOVA test* of the participant scores from the six played runs

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, which can be seen in **fig.** 5.4.



Pairwise Comparisons

Measure Points_Run

(I) Test	(J) Test	Mean Difference (I-J)	Std. Error	Sig.[b]	95% Confidence Interval for Difference[b]	
					Lower Bound	Upper Bound
1	2	663,333	1745,987	1,000	-4920,765	6247,431
	3	-996,667	1396,823	1,000	-5464,052	3470,719
	4	-1143,333	1431,466	1,000	-5721,517	3434,850
	5	-1783,333	1610,155	1,000	-6933,007	3366,341
	6	4390,000	1868,876	,388	-1587,127	10367,127
2	1	-663,333	1745,987	1,000	-6247,431	4920,765
	3	-1660,000	1449,357	1,000	-6295,400	2975,400
	4	-1806,667	1374,847	1,000	-6203,768	2590,435
	5	-2446,667	2112,289	1,000	-9202,288	4308,955
	6	3726,667	1661,117	,490	-1585,995	9039,328
3	1	996,667	1396,823	1,000	-3470,719	5464,052
	2	1660,000	1449,357	1,000	-2975,400	6295,400
	4	-146,667	1237,459	1,000	-4104,366	3811,032
	5	-786,667	1525,098	1,000	-5664,308	4090,975
	6	5386,667*	1606,758	,034	247,857	10525,477
4	1	1143,333	1431,466	1,000	-3434,850	5721,517
	2	1806,667	1374,847	1,000	-2590,435	6203,768
	3	146,667	1237,459	1,000	-3811,032	4104,366
	5	-640,000	1951,307	1,000	-6880,763	5600,763
	6	5533,333	1972,409	,133	-774,919	11841,586
5	1	1783,333	1610,155	1,000	-3366,341	6933,007
	2	2446,667	2112,289	1,000	-4308,955	9202,288
	3	786,667	1525,098	1,000	-4090,975	5664,308
	4	640,000	1951,307	1,000	-5600,763	6880,763
	6	6173,333*	1855,640	,036	238,537	12108,129
6	1	-4390,000	1868,876	,388	-10367,127	1587,127
	2	-3726,667	1661,117	,490	-9039,328	1585,995
	3	-5386,667*	1606,758	,034	-10525,477	-247,857
	4	-5533,333	1972,409	,133	-11841,586	774,919
	5	-6173,333*	1855,640	,036	-12108,129	-238,537

Based on estimated marginal means

*. The mean difference is significant at the ,05 level.

b. Adjustment for multiple comparisons: Bonferroni.

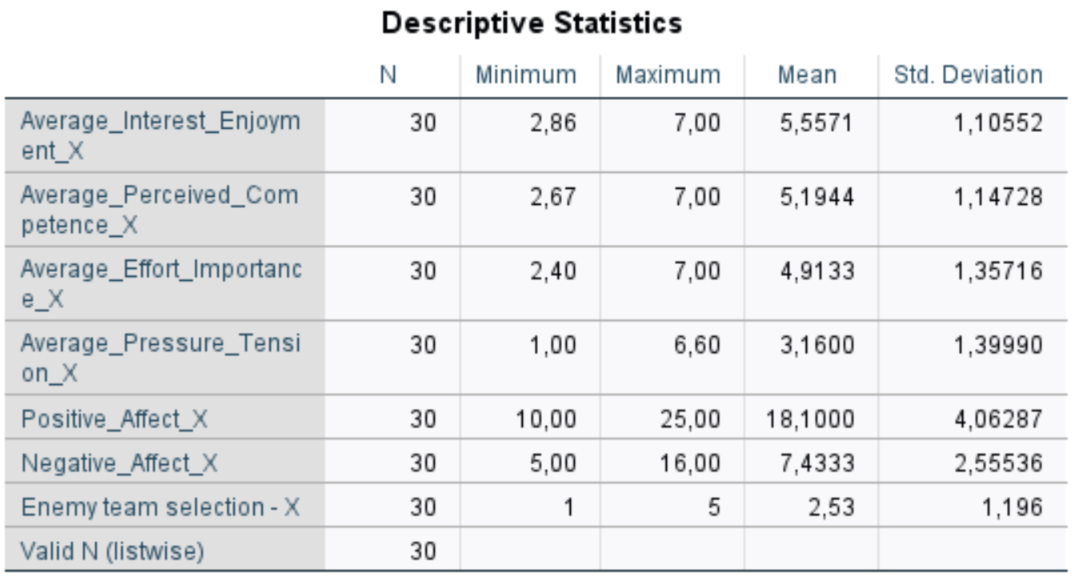
Figure 5.4: Pairwise comparison of the *Repeated Measures One-Way ANOVA test* of the participant scores

Looking at the figure above, we can see there are two pairs where the *p* value is below 0.05, meaning there is a statistically significant difference in pair 3 - 6 and in pair 5 - 6, which corresponds to 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*.

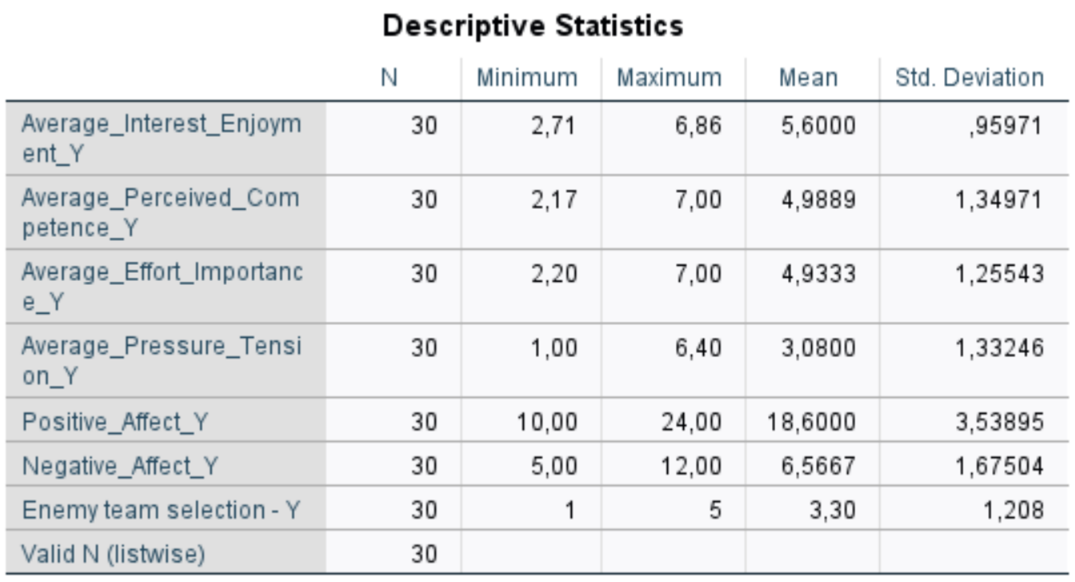
The collected descriptive statistics of these 7 variables for the 30 participants and for versions X, Y and Z can be seen in **fig.** 5.5, **fig.** 5.6, and **fig.** 5.7, respectively.



Descriptive Statistics

	N	Minimum	Maximum	Mean	Std. Deviation
Average_Interest_Enjoyment_X	30	2,86	7,00	5,5571	1,10552
Average_Perceived_Competence_X	30	2,67	7,00	5,1944	1,14728
Average_Effort_Importance_X	30	2,40	7,00	4,9133	1,35716
Average_Pressure_Tension_X	30	1,00	6,60	3,1600	1,39990
Positive_Affect_X	30	10,00	25,00	18,1000	4,06287
Negative_Affect_X	30	5,00	16,00	7,4333	2,55536
Enemy team selection - X	30	1	5	2,53	1,196
Valid N (listwise)	30				

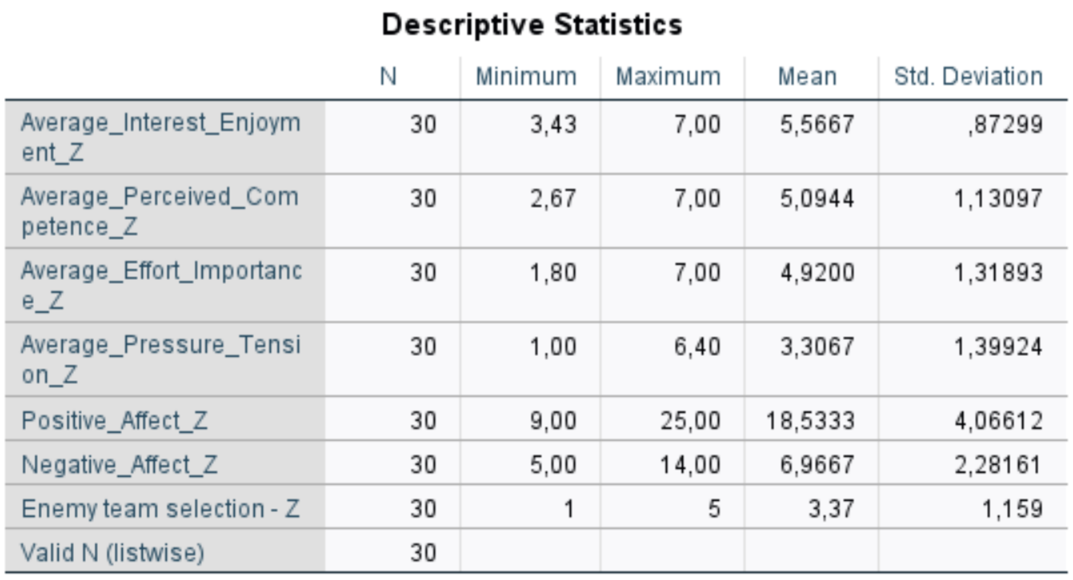
Figure 5.5: Participant scores of the 7 variables on version X



Descriptive Statistics

	N	Minimum	Maximum	Mean	Std. Deviation
Average_Interest_Enjoyment_Y	30	2,71	6,86	5,6000	,95971
Average_Perceived_Competence_Y	30	2,17	7,00	4,9889	1,34971
Average_Effort_Importance_Y	30	2,20	7,00	4,9333	1,25543
Average_Pressure_Tension_Y	30	1,00	6,40	3,0800	1,33246
Positive_Affect_Y	30	10,00	24,00	18,6000	3,53895
Negative_Affect_Y	30	5,00	12,00	6,5667	1,67504
Enemy team selection - Y	30	1	5	3,30	1,208
Valid N (listwise)	30				

Figure 5.6: Participant scores of the 7 variables on version Y



Descriptive Statistics

	N	Minimum	Maximum	Mean	Std. Deviation
Average_Interest_Enjoyment_Z	30	3,43	7,00	5,5667	,87299
Average_Perceived_Competence_Z	30	2,67	7,00	5,0944	1,13097
Average_Effort_Importance_Z	30	1,80	7,00	4,9200	1,31893
Average_Pressure_Tension_Z	30	1,00	6,40	3,3067	1,39924
Positive_Affect_Z	30	9,00	25,00	18,5333	4,06612
Negative_Affect_Z	30	5,00	14,00	6,9667	2,28161
Enemy team selection - Z	30	1	5	3,37	1,159
Valid N (listwise)	30				

Figure 5.7: Participant scores of the 7 variables on version Z

The analysis of each of the collected 7 variables that are depicted in the pictures above goes as follows:

- **Interest/Enjoyment:** Variable obtained from IMI, ranging from 1 to 7. Its values approximate to 5.6 in versions X, Y and Z, a value that is closer to the possible maximum, meaning the participants perceived being very interested and really enjoyed playing all three versions;

- **Perceived Competence:** Variable obtained from IMI, ranging from 1 to 7. Its values approximate to 5 in versions X, Y and Z, a value that is as close to the possible maximum as well as the average, meaning the participants perceived being competent playing all three versions;
- **Effort/Importance:** Variable obtained from IMI, ranging from 1 to 7. Its values approximate to 4.9 in versions X, Y and Z, a value that 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:** Variable obtained from IMI, ranging from 1 to 7. Its values approximate to 3.2 in versions X, Y and Z, a value that 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:** Variable obtained from PANAS, ranging from 5 to 25. Its values approximate to 18.4, a high achieved value that supports the participants to 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:** Variable obtained from PANAS, ranging from 5 to 25. Its values approximate to 7, a small achieved value that supports the participants to 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:** Variable obtained 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.

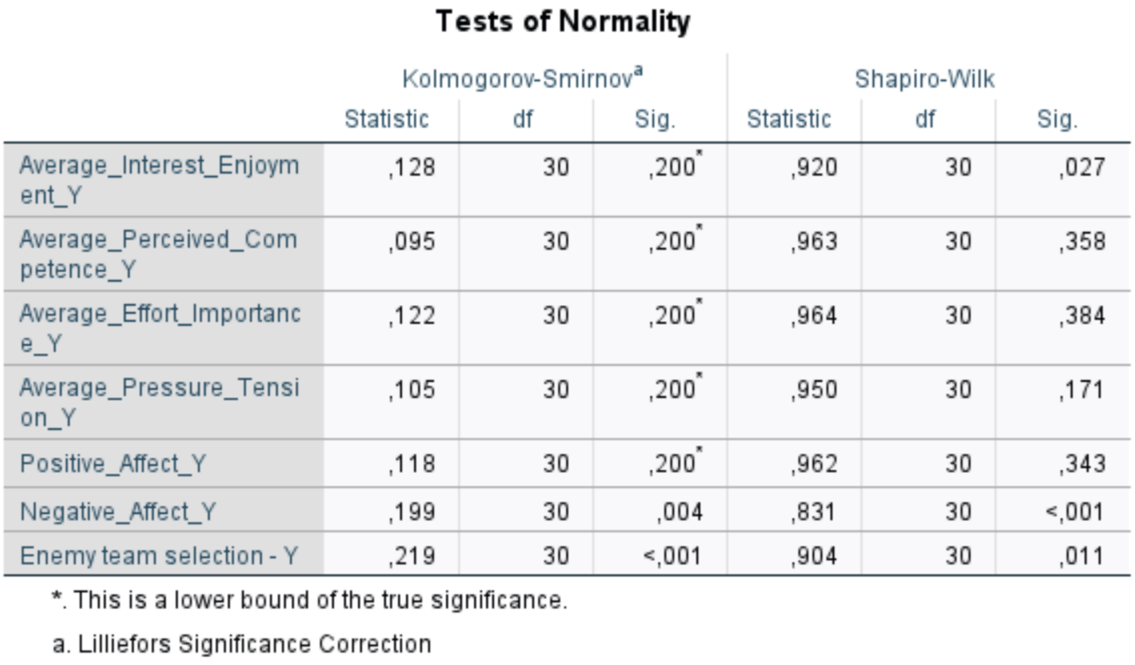
In order to test the variables for statistically significant difference, we used SPSS[2], a software platform, which offers advanced statistical analysis, a vast library of machine learning algorithms, text analysis, open-source extensibility, integration with big data and deployment into applications.

We started by testing each of these variables for normality in order to decide if we would use parametric or non-parametric tests. Since we have 3 different versions to test these variables on (Versions X, Y and Z), we needed to test each variable under each version. The *Shapiro-Wilk*[3] test of normality

[2]https://www.ibm.com/spss

[3]https://statistics.laerd.com/spss-tutorials/testing-for-normality-using-spss-statistics.php

was used, since the amount of samples (correspondent to the number of participants) was below 50. An example of the obtained results from the *Shapiro-Wilk* test of normality for the variables in version Y can be seen below, in **fig.** 5.8.



Tests of Normality

	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
Average_Interest_Enjoyment_Y	,128	30	,200*	,920	30	,027
Average_Perceived_Competence_Y	,095	30	,200*	,963	30	,358
Average_Effort_Importance_Y	,122	30	,200*	,964	30	,384
Average_Pressure_Tension_Y	,105	30	,200*	,950	30	,171
Positive_Affect_Y	,118	30	,200*	,962	30	,343
Negative_Affect_Y	,199	30	,004	,831	30	<,001
Enemy team selection - Y	,219	30	<,001	,904	30	,011

*. This is a lower bound of the true significance.

a. Lilliefors Significance Correction

Figure 5.8: *Shapiro-Wilk* test of normality obtained values from version Y

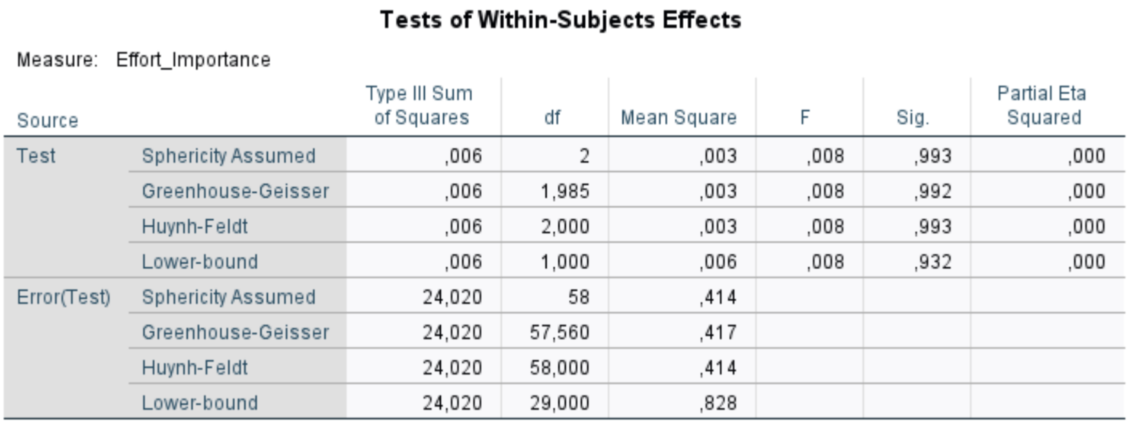
In this case, we can observe that the *Sig.* value of *Average_Interest_Enjoyment_Y*, *Negative_Affect_Y* and *Enemy_Team_Selection-Y* is below 0.05, which means the data significantly deviates from a normal distribution.

This test is done to all variables from the three versions. The obtained results are shown in **table** 5.1 below, with each cell representing *p*, the *Sig.* value. When this value is greater than 0.05, it means the data is normal and parametric tests can be done and, when the value is below 0.05, it means the data deviates from a normal distribution and non-parametric tests must be done.

Variable/Version	X	Y	Z
Interest/Enjoyment	0.089	0.027	0.435
Perceived Competence	0.476	0.358	0.332
Effort/Importance	0.177	0.384	0.174
Pressure/Tension	0.269	0.171	0.438
Positive Affect	0.313	0.343	0.390
Negative Affect	$<$0.001	$<$0.001	$<$0.001
Enemy Team Selection	0.005	0.011	0.003

Table 5.1: *P* values obtained as a result of the *Shapiro-Wilk* normality tests depicting each variable as parametric or non-parametric

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[4]. An example of this test on the *Average_Effort_Importance* can be seen on **fig.** 5.9.


Tests of Within-Subjects Effects

Measure: Effort_Importance

Source		Type III Sum of Squares	df	Mean Square	F	Sig.	Partial Eta Squared
Test	Sphericity Assumed	,006	2	,003	,008	,993	,000
	Greenhouse-Geisser	,006	1,985	,003	,008	,992	,000
	Huynh-Feldt	,006	2,000	,003	,008	,993	,000
	Lower-bound	,006	1,000	,006	,008	,932	,000
Error(Test)	Sphericity Assumed	24,020	58	,414			
	Greenhouse-Geisser	24,020	57,560	,417			
	Huynh-Feldt	24,020	58,000	,414			
	Lower-bound	24,020	29,000	,828			

Figure 5.9: *One-Way ANOVA with Repeated Measures test* on *Average_Effort_Importance* variable

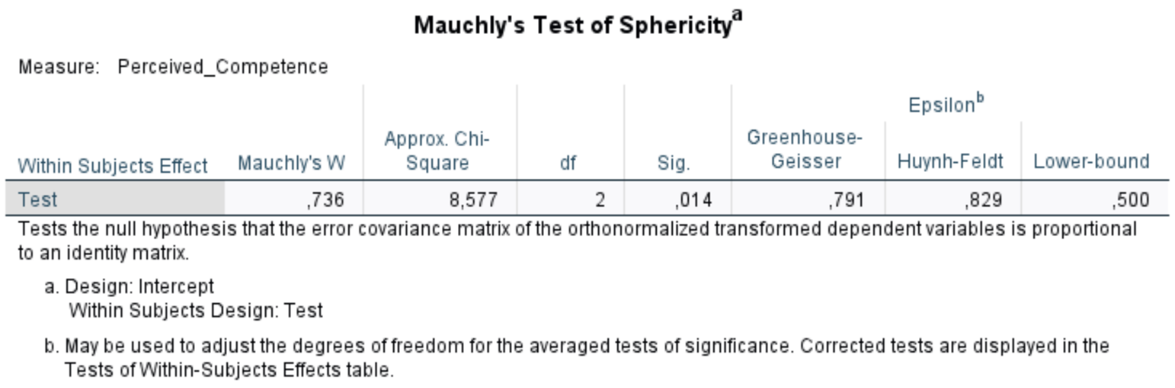
In the figure above, we can see that a Repeated Measures One-Way ANOVA test with a Greenhouse-Geisser correction determined that there is no statistically significant difference on the perceived effort/importance between the three versions (F(1.985, 57.560) = 0.008, p = 0.992), since p is greater than 0.05.

The same conclusion was obtained when performing the test to *Average_Pressure_Tension*, *Positive_Affect* and *Average_Perceived_Competence*, 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) =

[4]https://statistics.laerd.com/spss-tutorials/one-way-anova-repeated-measures-using-spss-statistics.php

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[5] ($\chi^2(2)$ = 8.577, p = 0.014), as depicted on **fig.** 5.10, which means p is below 0.05, thus violating sphericity.



Mauchly's Test of Sphericity[a]

Measure: Perceived_Competence

					Epsilon[b]		
Within Subjects Effect	Mauchly's W	Approx. Chi-Square	df	Sig.	Greenhouse-Geisser	Huynh-Feldt	Lower-bound
Test	,736	8,577	2	,014	,791	,829	,500

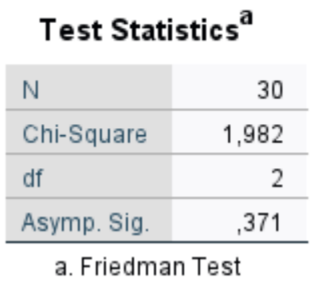
Tests the null hypothesis that the error covariance matrix of the orthonormalized transformed dependent variables is proportional to an identity matrix.

a. Design: Intercept
Within Subjects Design: Test

b. May be used to adjust the degrees of freedom for the averaged tests of significance. Corrected tests are displayed in the Tests of Within-Subjects Effects table.

Figure 5.10: *Mauchly's Test of Sphericity* on the *Average_Perceived_Competence* variable

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[6]. 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. An example of this test on *Average_Perceived_Competence* is depicted on **fig.** 5.11.



Test Statistics[a]

N	30
Chi-Square	1,982
df	2
Asymp. Sig.	,371

a. Friedman Test

Figure 5.11: *Friedman Test* on the *Average_Perceived_Competence* variable

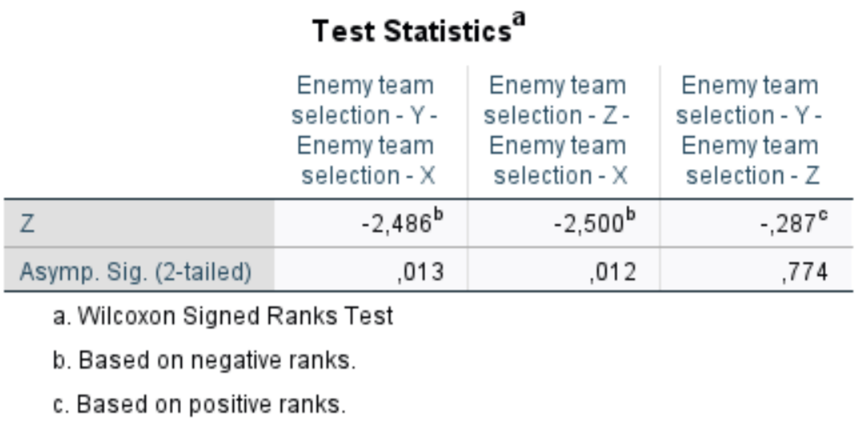
The figure above confirms the result we previously obtained regarding the *Average_Perceived_Competence* variable, which fails the Friedman Test ($\chi^2(2)$ = 1.982, p = 0.371), since p is greater than 0.05 and thus meaning there is no statistically significant difference on the perceived competence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

The same conclusion was obtained when performing the test to *Average_Interest_Enjoyment* and

[5] https://statistics.laerd.com/statistical-guides/sphericity-statistical-guide.php
[6] https://statistics.laerd.com/spss-tutorials/testing-for-normality-using-spss-statistics.php

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[7], which can be observed on **fig.** 5.12.



Test Statistics[a]

	Enemy team selection - Y - Enemy team selection - X	Enemy team selection - Z - Enemy team selection - X	Enemy team selection - Y - Enemy team selection - Z
Z	-2,486[b]	-2,500[b]	-,287[c]
Asymp. Sig. (2-tailed)	,013	,012	,774

a. Wilcoxon Signed Ranks Test

b. Based on negative ranks.

c. Based on positive ranks.

Figure 5.12: *Wilcoxon Signed-Rank Test* on the *Enemy_Team_Selection* variable

With the obtained results from the Wilcoxon Signed-Rank 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.** 5.13.

[7]https://statistics.laerd.com/spss-tutorials/wilcoxon-signed-rank-test-using-spss-statistics.php

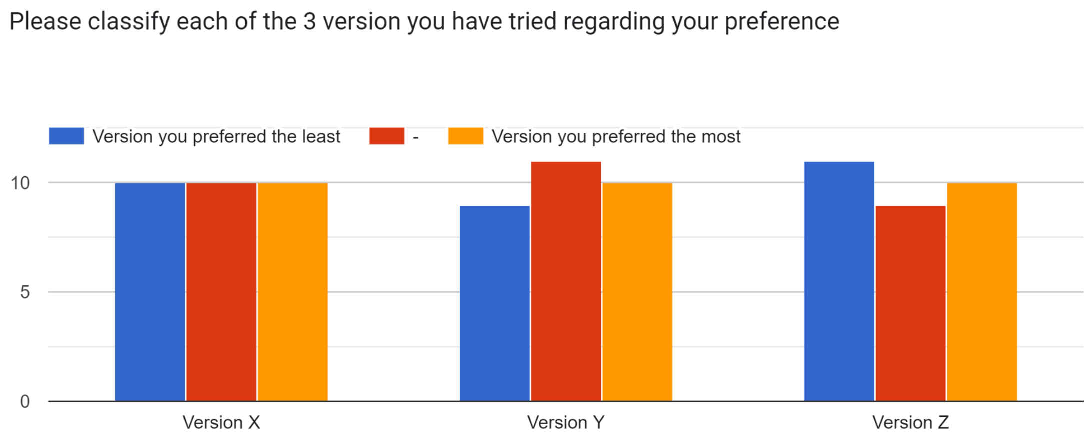


Figure 5.13: Graph bar of the participants' most and least preferred versions

In the last 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 1 less vote than version X and 2 fewer votes than version Y, represented by the blue bar. The red bar represents the version that was neither the one that was most nor least preferred.

5.3 Discussion

In the stated hypothesis on section 1.3 we proposed the player would have a preference order with a version where challenges are adapted to an Elo-based model that measures the player's skill (A version) coming first, followed by a version where challenges are ordered based on their difficulty (NA version) and, lastly, a version where challenges are randomly selected.

The first step for testing these hypothesis is to look at the bar graph on fig. 5.13, 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.

The results from the four measured dimensions of IMI support that players have a very similar experience while playing the three versions: participants were very interested and quite enjoyed playing the game; participants perceived to be competent; participants felt this was an important task and made an effort to complete it; players were not that pressured and did not feel that much tension while playing.

Regarding the participants' affectivity while playing the game, although there is only a minimal difference, version Y, which corresponds to the NA version, is the one with a higher value for positive affect and lower value for negative affect. Version X, which corresponds to the R version, is the one with lower value for positive affect and higher value for negative affect. This would mean that the players' order for more positive sensations and emotions would lean to, from most to least, the NA version, followed by the A version and lastly the R version.

We need, however, to test these results and verify if there is a statistically significant difference in order to support this premise. Upon performing the tests, we conclude there is no statistically significant difference in any of the six aforementioned variables, which means there is no difference in perceived experience between the three versions.

There is, however, 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.

Concluding, we are not able to define which version the participants preferred playing, thus not being able to prove or refute our stated hypothesis.

6

Conclusion

Contents

In this section, we conclude our work, by stating some final remarks, while looking at all that was done and what could be done in possible future work.

6.1 Conclusions

In this thesis document, 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.

A final consideration to take is that, although different players have different skills and different playing experiences, our developed heuristic was able to classify the difficulty of each challenge accurately, thus allowing us to create easier or harder challenges at the designer will. This heuristic provided a better playing experience for the players by being able to select the intended challenges with minimal resources and time.

6.2 Future Work

To conclude this document, some topics follow that may be interesting to investigate:

- Apply the work that was described in this document to a different type of game, where there is more variety on what type of challenges the player can face and compare results;
- Have a higher number of participants (> 50), which would allow us divide them in subgroups according to their playing experience and compare results;
- Tweaking some of the variable used in this work, such as a bigger number of pre-generated challenges (more than 1000) and a bigger number for the bootstrap value of the A version (more than 20), which would allow for a wider challenge variety, and compare results;
- Have an ANN take place of the challenge selection heuristic and compare its accuracy;
- Having focus groups to perform the test. Separate groups by their playing experience or by their type[1][2], since these differences amongst the players might affect the perception on their experience, and compare results;

[1] https://www.interaction-design.org/literature/article/bartle-s-player-types-for-gamification
[2] https://blog.brainhex.com/

Bibliography

[1] G. Smith, “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*, 2014, pp. 917–926.

[2] J. F. L. Pardal, “Holiday knight: a videogame with skill-based challenge generation,” Master’s thesis, Instituto Superior Técnico, 2019.

[3] M. Csikszentmihalyi and M. Csikzentmihaly, *Flow: The psychology of optimal experience*. Harper & Row New York, 1990, vol. 1990.

[4] G. Chanel, C. Rebetez, M. Bétrancourt, and T. Pun, “Boredom, engagement and anxiety as indicators for adaptation to difficulty in games,” in *Proceedings of the 12th international conference on Entertainment and media in the ubiquitous era*, 2008, pp. 13–17.

[5] Y. L. Hanin, “Emotions and athletic performance: Individual zones of optimal functioning model.” *D. Smith & M. Bar-Eli (Eds.)*, pp. 55–73, 2007.

[6] N. Shaker, J. Togelius, and M. J. Nelson, *Procedural content generation in games*. Springer, 2016, ch. Introduction, pp. 1–15.

[7] C. Browne and F. Maire, “Evolutionary game design,” *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 2, no. 1, pp. 1–16, 2010.

[8] G. Smith, J. Whitehead, and M. Mateas, “Tanagra: A mixed-initiative level design tool,” in *Proceedings of the Fifth International Conference on the Foundations of Digital Games*, 2010, pp. 209–216.

[9] N. A. Barriga, “A short introduction to procedural content generation algorithms for videogames,” *International Journal on Artificial Intelligence Tools*, vol. 28, no. 02, p. 1930001, 2019.

[10] A. Doull, “The death of the level designer,” *URL http://pcg. wikidot. com/the-death-of-the-level-designer*, 2008.

[11] J. Togelius, G. N. Yannakakis, K. O. Stanley, and C. Browne, “Search-based procedural content generation,” in *European Conference on the Applications of Evolutionary Computation*. Springer, 2010, pp. 141–150.

[12] G. N. Yannakakis and J. Togelius, “Experience-driven procedural content generation,” in *2015 International Conference on Affective Computing and Intelligent Interaction (ACII)*. IEEE, 2015, pp. 519–525.

[13] M.-V. Aponte, G. Levieux, and S. Natkin, “Measuring the level of difficulty in single player video games,” *Entertainment Computing*, vol. 2, no. 4, pp. 205–213, 2011.

[14] A. Sarkar and S. Cooper, “Transforming game difficulty curves using function composition,” in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, 2019, pp. 1–7.

[15] B. Horn, A. K. Hoover, Y. Folajimi, J. Barnes, C. Harteveld, and G. Smith, “Ai-assisted analysis of player strategy across level progressions in a puzzle game,” in *Proceedings of the 12th International Conference on the Foundations of Digital Games*, 2017, pp. 1–10.

[16] T. Allart, G. Levieux, M. Pierfitte, A. Guilloux, and S. Natkin, “Difficulty influence on motivation over time in video games using survival analysis,” in *Proceedings of the 12th International Conference on the Foundations of Digital Games*, 2017, pp. 1–6.

[17] R. Hunicke, “The case for dynamic difficulty adjustment in games,” in *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology*, 2005, pp. 429–433.

[18] D. Ang and A. Mitchell, “Comparing effects of dynamic difficulty adjustment systems on video game experience,” in *Proceedings of the Annual Symposium on Computer-Human Interaction in Play*, 2017, pp. 317–327.

[19] J. C. B. P. Catarino, “Skill-based progression model for smash time,” Master’s thesis, Instituto Superior Técnico, 2018.

[20] F. F. N. P. Bicho, “Multi-dimensional player skill progression modeling for procedural content generation,” Master’s thesis, Instituto Superior Técnico, 2018.

[21] F. S. Mestre, “Multi-dimensional elo-based challenge progression for single-player games,” Master’s thesis, Instituto Superior Técnico, 2020.

[22] J. Catarino and C. Martinho, “Procedural progression model for smash time,” in *2019 IEEE Conference on Games (CoG)*, 2019, pp. 1–8.

[23] F. Bicho and C. Martinho, “Multi-dimensional player skill progression modelling for procedural content generation,” in *Proceedings of the 13th International Conference on the Foundations of Digital Games*, ser. FDG ’18. New York, NY, USA: Association for Computing Machinery, 2018. [Online]. Available: https://doi.org/10.1145/3235765.3235774

[24] B. Morrison, *Comparing elo, glicko, irt, and bayesian irt statistical models for educational and gaming data*. University of Arkansas, 2019.

[25] N. E. Newton-Fisher, “Modeling social dominance: Elo-ratings, prior history, and the intensity of aggression,” *International journal of primatology*, vol. 38, no. 3, pp. 427–447, 2017.

[26] C. S. Tsang, H. Y. Ngan, and G. K. Pang, “Fabric inspection based on the elo rating method,” *Pattern Recognition*, vol. 51, pp. 378–394, 2016.

[27] M. J. Brinkhuis and G. Maris, “Dynamic parameter estimation in student monitoring systems,” *Measurement and Research Department Reports (Rep. No. 2009-1). Arnhem: Cito*, 2009.

[28] R. Pelánek, “Applications of the elo rating system in adaptive educational systems,” *Computers & Education*, vol. 98, pp. 169–179, 2016.

[29] W. A. IJsselsteijn, Y. A. de Kort, and K. Poels, “The game experience questionnaire. eindhoven: Technische universiteit eindhoven,” 2013.

[30] A. Hughes and P. C. Kendall, “Psychometric properties of the positive and negative affect scale for children (panas-c) in children with anxiety disorders,” *Child Psychiatry and Human Development*, vol. 40, pp. 343–352, 2009.

[31] K. Kercher, “Assessing subjective well-being in the old-old: The panas as a measure of orthogonal dimensions of positive and negative affect,” *Research on Aging*, vol. 14, no. 2, pp. 131–168, 1992.

[32] E. R. Thompson, “Development and validation of an internationally reliable short-form of the positive and negative affect schedule (panas),” *Journal of Cross-Cultural Psychology*, vol. 38, no. 2, pp. 227–242, 2007.

[33] D. Watson and L. A. Clark, “The panas-x: Manual for the positive and negative affect schedule - expanded form,” p. 24 pages, 1994.

A

Final Questionnaire

Holiday Knight Player Questionnaire

Thank you for trying out Holiday Knight and being a part of this study!

I am developing 3 versions of the Holiday Knight video game for my Master's degree and would like to know how you felt while playing and your opinion regarding each of the 3 versions.

After answering a few questions about yourself, I would like to ask you to try each version of the game and answer a short questionnaire. After trying all the versions, one final questionnaire will be prompted. The whole experiment should last a maximum of 45 minutes.

Finally, I would like to inform you that your participation on this activity is completely voluntary and you are free to interrupt it at any point in time. If you have any questions, you can contact me via email: jorge.costa.nunes@tecnico.ulisboa.pt.

By starting this survey, you are consenting to the last depicted point.

Thank you and have fun!

*Required

About You

In the following section, I will ask you some questions about yourself and your relationship with video games.

1. How old are you? *

____

2. What is your gender? *

Mark only one oval.

- ◯ Female
- ◯ Male
- ◯ Other
- ◯ Prefer not to say

3. How often do you play video games? *

Mark only one oval.

- ⬭ I make some time in my schedule to play video games.
- ⬭ I play video games occasionally when the opportunity presents itself.
- ⬭ I do not play video games.

4. Are you familiar with games in which the protagonist combats a large number of enemies by shooting at them while dodging their fire, such as Enter the Gungeon, Binding of Isaac, Cuphead or Nuclear Throne? *

Mark only one oval.

- ⬭ I enjoy these games and have played/watched others play them multiple times.
- ⬭ I played/watched others play them enough to understand I do not appreciate them.
- ⬭ I am not familiar with these games and/or have no formed opinion on them.

[TO BE SELECTED BY THE OVERSEER]

You should now reach out to the overseer in order to progress to continue the experiment.

5. [TO BE SELECTED BY THE OVERSEER] *

Mark only one oval.

- ⬭ Move to "Version X" *Skip to question 6*
- ⬭ Move to "Version Y" *Skip to question 10*
- ⬭ Move to "Version Z" *Skip to question 14*

Holiday Knight - Version X

The following section should only be filled after completing the developed Version X of Holiday Knight.

6. Please indicate how you felt while playing the game for each of the items, on the following scale: *

Mark only one oval per row.

	1 - Not at all true	2	3	4 - Somewhat true	5	6	7 - Very true
I enjoyed doing the game very much.							
This game was fun to play.							
I thought this was a boring game.							
This game did not hold my attention at all.							
I would describe this game as very interesting.							
I thought this game was quite enjoyable.							
While I was playing this game, I was thinking about how much I enjoyed it.							
I think I was pretty good at this game.							

I think I did pretty well at this game, compared to other people.	○	○	○	○	○	○	○
After playing this game for a while, I felt pretty competent.	○	○	○	○	○	○	○
I am satisfied with my performance at this game.	○	○	○	○	○	○	○
I was pretty skilled at this game.	○	○	○	○	○	○	○
This was a game that I couldn't play very well	○	○	○	○	○	○	○
I put a lot of effort into playing this game.	○	○	○	○	○	○	○
I didn't try very hard to do well at this game.	○	○	○	○	○	○	○
I tried very hard on this game.	○	○	○	○	○	○	○
It was important to me to do well at this game.	○	○	○	○	○	○	○
I didn't put much energy	○	○	○	○	○	○	○

into playing this game.							
I did not feel nervous at all while playing this game.	◯	◯	◯	◯	◯	◯	◯
I felt very tense while playing this game.	◯	◯	◯	◯	◯	◯	◯
I was very relaxed while playing the game.	◯	◯	◯	◯	◯	◯	◯
I was anxious while playing this game.	◯	◯	◯	◯	◯	◯	◯
I felt pressured while playing this game.	◯	◯	◯	◯	◯	◯	◯

7. Please indicate the extent of what you felt while playing the game. *

Mark only one oval per row.

	Very slightly or not at all	A little	Moderately	Quite a bit	Extremely
Active	◯	◯	◯	◯	◯
Attentive	◯	◯	◯	◯	◯
Alert	◯	◯	◯	◯	◯
Determined	◯	◯	◯	◯	◯
Inspired	◯	◯	◯	◯	◯
Hostile	◯	◯	◯	◯	◯
Ashamed	◯	◯	◯	◯	◯
Upset	◯	◯	◯	◯	◯
Afraid	◯	◯	◯	◯	◯
Nervous	◯	◯	◯	◯	◯

8. In what way did you feel the enemy teams were selected in Version X?

Mark only one oval.

	1	2	3	4	5	
Completely Random	◯	◯	◯	◯	◯	Completely Deterministic

[TO BE SELECTED BY THE OVERSEER]

You should now reach out to the overseer in order to progress to continue the experiment.

17. [TO BE SELECTED BY THE OVERSEER] *

Mark only one oval.

- Move to "Version X" *Skip to question 6*
- Move to "Version Y" *Skip to question 10*
- Move to "Final Questionnaire" *Skip to question 18*

Final Questionnaire

In this last section I will be asking your general opinion regarding the 3 developed versions of the Holiday Knight video game.

18. Please classify each of the 3 version you have tried regarding your preference *

Mark only one oval per row.

	Version you preferred the least	-	Version you preferred the most
Version X			
Version Y			
Version Z			

19. Why did you choose the "Version you preferred the most" in the last question as your preferred version? *

20. Why did you choose the "Version you preferred the least" in the last question as your least preferred version? *

Thank you!

Thank you so much for participating in this survey! I really appreciate it! Hope you enjoyed playing the game! :)