FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Towards Machine Learning in Game Adaptivity

Francisco Lopes

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: João Jacob

Second Supervisor: Zafeiris Kokkinogenis

October 29, 2020

Towards Machine Learning in Game Adaptivity

Francisco Lopes

Mestrado Integrado em Engenharia Informática e Computação

October 29, 2020

Abstract

Computer games, along with television series, movies and books, are one of the options we have to entertain ourselves. Exclusive to the computer game world is the accessibility, difficulty and replayability of them. Due to these two factors, it is important to find out how to design a game so that those factors can be optimized for the most accessible and engaging experience. Various techniques have been employed to maximize these two factors but the one focused on this dissertation is Game Adaptivity.

Since every player has different opinions on the best gameplay and pacing, as well as being at different skill levels ultimately impacting their enjoyment of the game experience, it is important to find a way to adapt the game to a particular player's skill level, preferred game content, preferred gameplay, among others. Game adaptivity is one of the answers to this problem and can be used to fine tune the difficulty for the player, so that it is not too challenging or too difficult. It can also be used to adapt the gameplay to a player's inclinations so that it offers a more engaging game experience.

Game adaptivity is divided into offline and online game adaptivity. Whereas the offline version focuses on procedural generation before play, the online version focuses on real-time techniques that can be used to adapt the game experience. Usually this is done by modifying small components of the game such as specific game elements or tweaking Non-Playable Character's (NPC) Artificial Intelligence (AI). In this work, a novel approach to Game Adaptivity is going to be researched by using Machine Learning (ML) to automatically adapt game content, specifically level design, so that the gaming experience can be improved. As input for the Machine Learning process, the performance (relative or absolute) of previous players is going to be used.

Using a custom computer game that has tunable parameters so that the ML process can change the game, a non-adaptive version of the game and an adaptive version of the game were developed. Using a k-nearest neighbors algorithm, the current player performance was matched to previous player performances, and then the procedural generation tweaked to be similar to the closest matched player. The idea being to reproduce the same experience on the new player in hopes of improving their experience, since the previous player performances were composed of only people that liked the game. For evaluation, a group of users tested the non-adaptive version and another the adaptive version. Through their user logs of the play sessions, and the Game Experience Questionnaire, the game experience was analyzed to check whether the experience was indeed improved between the sets of testers. The results obtained show that users liked both versions, but the adaptive version had no significant impact, negative or positive, on their game experience, concluding that the method chosen was not able to achieve its purpose. However, the adaptive version did result in users feeling more competent at the game due to lessened challenge.

Keywords: computer games, player centered design, dynamic difficulty adjustment, machine learning, game adaptivity, knn

Acknowledgments

I would like to thank the supervisors for their help along the process, specially their patience given the process dragged a bit.

To my friends Zé and Inês, a very warm thank you. Pairing an healthy dose of unhealthy foods with 80's movies and good companionship was never better.

My parents waited a long time for me to conclude my studies, and did everything they could during those long years to make my life easier. Still, they put up with my being difficult, so to them I extend my eternal gratitude. I know their support will not end here and I will do better in showing appreciation for their actions. Thank you.

Last but not least, my girlfriend Matilde, who endured quarantine with me and also had to put up with me during the whole thesis process, both hers and mine. She was there through the whole process, quite literally, and was a major help in making me reach the end of this road. I love you and thank you for your patience and love. Here's to new beginnings with my favorite person!

Contents

List of Figures

List of Tables

Abbreviations

AI	Artificial Intelligence
CSV	Comma-Separated Values
GEQ	Game Experience Questionnaire
GUID	Globally Unique Identifier
k-NN	k-Nearest Neighbors
ML	Machine Learning
NPC	Non-Player Character
OS	Operating System
RL	Reinforcement Learning

Chapter 1

Introduction

Gaming is one of the largest sectors of the entertainment world spanning several genres and types of games such as serious games, simulation games and casual games. This variety leads to the industry appealing to a large userbase.

To further appeal to different kinds of players, the industry has long used difficulty levels and related settings to customize the gaming experience. This can lead to an experience that is more in sync with the player's expectations but relies on the player's self-profiling to actually tweak the settings. Problems may arise from this approach since the players might not know how to profile themselves or they may not think that they fit in any of the prescribed options the game offers [1].

To counteract the need for players to profile themselves and associated shortcomings, games have attempted to profile the players automatically and using that profile, adapt the game around the player. Examples of such adaptivity have been seen in Remedy Entertainment's Max Payne [2] which tuned the aim assist according to how well the player's accuracy was when hitting enemies [1], or in Valve's Left 4 Dead series which feature an AI Director that decides certain events, pickup item spawns and enemy spawns as to keep the group challenged just the right amount, to provide the best experience [3].

Several techniques have been explored for adapting a game's content to a specific player such as procedural generation used heavily in the game Spore [4] to construct a different world for every player, or on a more academic level to generate a Platformer's levels [5], using agent bidding concepts to improve serious games [6] or improving the Artificial Intelligence (AI) of a Real Time Strategy game [7]. Aligned with those works, this work intends to advance the field of Game Adaptivity, using Machine Learning techniques as the basis for adaptivity.

1.1 Context and Motivation

As previously mentioned in section 1, the video game industry is one of the largest sectors of the entertainment world, valued at the multi billion dollar level as can be seen in Figure 1.1. Being

such a large industry, there is a continuous push to attract new customers which leads to the problem of figuring out the best methods to appeal to new customers. Every player has their own opinion on what the best experience is for them, that is, what is the best gameplay, best pacing, best story, best difficulty level, for that particular player.

Games already feature different themes, different target audiences, different age ratings, different genres, customizable elements so that the player may personalize the experience to their tastes. However, to further meet the player's expectations and desires, it becomes important to find a way to adapt the experience to a particular player, without requiring the player to self-profile. This is the motivation for the work, finding new ways through which games can adapt themselves to the individual player's expectations and desires, a process studied by the field of Game Adaptivity.

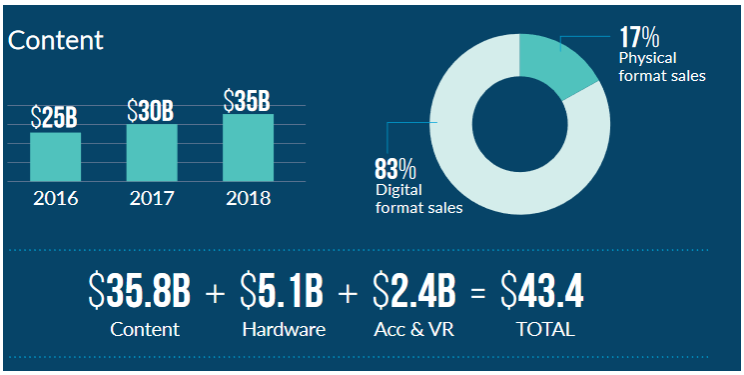


Figure 1.1: Sales figures of content, hardware and accessories in the video game industry [8]

1.2 Objectives

To make a contribution to the field of Game Adaptivity, several objectives were laid out for this work to ensure that it has a clear goal and well-defined scope. They are as follows:

- Research the field of Game Adaptivity to find methods of matching the player's preferences, what content can be adapted and what metrics can be used to optimize during the adaptation process.
- Research Machine Learning (ML) techniques that might work for adapting a game.
- Research possible Player Modeling techniques to use as input for the ML process.
- Design a ML-based framework for Game Adaptivity.
- Implement a prototype of the framework using a specific game as basis.
- Evaluate the prototype.

These objectives roughly map to the various stages of the work, the first three are important research for the state of the art review and the last ones concern the implementation and validation of a proposed solution.

1.3 Document Structure

This document is divided into several chapters for easier reading. Chapter 1, which is the current one, introduces the problem, contextualizes it and provides motivation and objectives for the work. The following chapter, 2, provides a state of the art review regarding the fields of Machine Learning and Game Adaptivity since they are the focus of the work and are needed in order to contribute to the field of Game Adaptivity. Chapter 3 describes the proposed solution that was implemented for the prototype. For that, the concept of an adaptive game framework is explained along with the decisions that were made in order to implement a specific adaptive game with a defined genre and type of content to adapt. The architecture of the proposed solution and the tools and technologies used are also described. Finally, a section detailing the implemented game and the creation of the adaptive version follow. Chapter 4 details how user tests were performed and what was gathered from them. Afterwards, the results are shown and discussed. To finish up, chapter 5 concludes the document and what was learned from the process, along with possible future work.

Chapter 2

Game Adaptivity through Machine Learning: a review

The purpose of this chapter is to ensure that the contribution of the work is novel and has a well-defined scope. To that end, a review of the fields of Machine Learning and Game Adaptivity is made for a better understanding of what has been done and what steps can be reused to reach new developments in the field of Game Adaptivity.

The state of the art review begins with section 2.1 which reviews current techniques in the field of Machine Learning. Afterwards, section 2.2 reviews the field of Game Adaptivity by defining it, listing what content can be adapted, offline and online adaptivity, and a section on Player Modeling. The following section, 2.3, reviews related works that have used ML techniques for Game Adaptivity. Finally, section 2.4 provides a summary of the state of the art review to highlight its key points.

2.1 Machine Learning

A subset of the field of Artificial Intelligence, Machine Learning tries to make computers learn by themselves, usually improving the efficiency of a given program during execution. This relies on inference and generalization of the available data to provide a solution [9].

While not the contribution field of this work, Machine Learning will be used as a tool to try and improve the field of Game Adaptivity and as such, a review of the ML field is required to ensure that the available techniques were properly researched. The purpose of ML in this work is to automatically adapt a video game so that it better fits the player. The perceived quality of the adaptation, which can be measured in multiple ways, works as the metric to optimize by the chosen ML method.

This section of the state of the art review will describe and analyze techniques currently used in the field of Machine Learning. The first section, 2.1.1, concerns Reinforcement Learning (RL) and

Transfer Learning applied to RL. Section 2.1.2 follows with the technique of Imitation Learning. Sections 2.1.3 and 2.1.4 concern k-nearest neighbors (k-NN) and recommender systems respectively. To conclude, section 2.1.5 provides a summary of the Machine Learning section of the state of the art review.

2.1.1 Reinforcement Learning

Reinforcement Learning has the goal of mapping states to actions with the objective of maximizing a scalar reward. The learning agent must try out (explore) the possible actions so that it can figure out which actions yield the highest reward. To do this, given an environment, the agent receives sensory information about its surroundings and must then choose an action to take. From this action it receives a reward signal. The goal being to maximize the cumulative reward received. The final mapping of states to best actions to take is called the policy and tells the agent which action to take given the current situation [10]. An overview of the concept of Reinforcement Learning can be seen in Figure 2.1. Notable implementations of Reinforcement Learning in academia include: Temporal Difference [11], Q-Learning [12] and Deep Reinforcement Learning [13].

Figure 2.1: Overview of Reinforcement Learning, the goal is maximization of the reward [10]

2.1.1.1 Transfer Learning

While reinforcement learning and other techniques start by learning from scratch in hopes of arriving at a solution, Transfer Learning attempts to transfer knowledge between tasks in hopes of having a positive effect on task performance. This means that given a set of tasks, the training experience for each task and performance measures for each task, raising the number of tasks and training experience of each task should improve performance if transfer is occurring between tasks [14]. In other words, knowledge from previously solving some task should be transferable to a new task that shares similarities.

Transfer learning can also be applied to RL techniques. While RL techniques already arrive at near-optimal solutions without any supervision, doing so might require a large number of samples which can be prohibitive when dealing with real-world scenarios. The aim of transfer learning

when applied to RL is then to transfer knowledge from learning a set of similar source tasks by biasing (to good hypotheses) the learning process when learning a new task [15].

The advantages that might occur when applying the concept of transfer learning to RL are [16]:

- **jumpstart** - transfer from a source task improves the initial performance of an agent in a target task.
- **asymptotic performance** - transfer improves the final performance of an agent in the target task.
- **total reward** - transfer improves the total accumulated reward.

Examples of implementation of transfer learning in the domain of RL include: a method called Progressive RL [17], transfer learning in variable-reward hierarchical RL [18], transfer learning through partial policy recycling [19], and graph-based domain mapping for transfer learning [20].

2.1.2 Imitation Learning

Humans have the ability to learn by mimicking at a very early stage. For example a study proved that newborns were capable of learning head gestures by imitation [21]. This concept of human imitation is the basis for Imitation Learning. Adapting this concept of human mimicry to the ML domain, an expert demonstrates the desired behavior and an agent tries to emulate that behavior [22].

Imitation learning has the advantage of circumventing the curse of dimensionality, since it reduces the state-action space due to expert following, and the advantage of avoiding potential consequences of trial-and-error methods, such as RL [23].

The workflow shown in Figure 2.2 represents the imitation learning process as proposed by Hussein [22]. Initially, there is a phase of capturing actions from the expert so that the agent can learn from those actions. Then, the actions are processed to extract features describing the state and environment of the expert. These features are used to learn a policy that mimics the expert behavior. Finally, the policy can be optimized by having the agent act out the policy and refine it based on its performance.

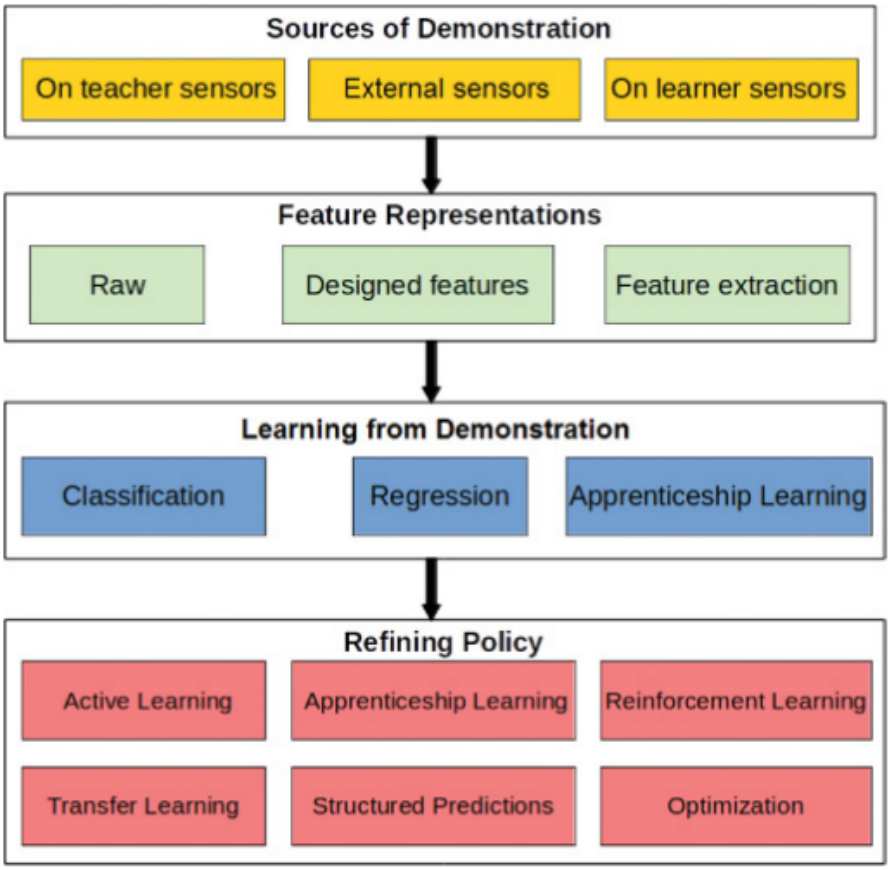


Figure 2.2: Imitation learning flowchart as proposed by Hussein [22]

Examples of implementation of imitation learning include Inverse Reinforcement Learning, which bears close resemblance to reinforcement learning as detailed in section 2.1.1. In inverse reinforcement learning it is assumed that the provided expert's policy is optimal in regards to an unknown reward function and the agent tries to learn said reward function. Afterwards, using a regular RL method the agent can optimize its policy using the reward function that it learned, hopefully arriving at a policy that is close to the expert's policy [24]. In another implementation example, Active Imitation Learning, the agent can query the expert about specific states in regards to the correct action to take with the goal of learning a policy that is close to the expert's policy while minimizing the number of queries to the expert. This method does not require knowing the expert's full trajectory [25].

2.1.3 k-Nearest Neighbors

The nearest neighbors algorithm serves the purpose of classifying a new sample point given a set of previously classified points. Given the set of previously classified points, the algorithm measures the distance (euclidean, Manhattan, others) from them to the new sample point. It is assumed that the new sample point will have the same classification as its nearest neighbors. The k value

indicates how many nearest neighbors should decide on the classification of the new sample point. Ultimately, classification is usually done by majority voting [26].

The k-NN algorithm has a few drawbacks, such as weighing the k neighbors equally through majority voting disregarding the possibility that a given nearest neighbor might have better classification ability. Another drawback is that the algorithm does not include a consideration for spatial distribution, caring only about simple distance calculations. Finally, the value of k is highly responsible for the accuracy that the algorithm will achieve, making it hard to determine how much it should be without trial and error. Newer developments in the area have lead to some of these problems being fixed, namely the majority voting being refined with a second majority class to serve as discrimination class, and adaptive k methods that determine it in an automatic way according to some rule instead of leaving it up to human guesswork [27].

This algorithm could be used to match user performances in order to classify them as being a particular type of player. This in turn could be used to tune the game in a way that works better for that type of player.

2.1.4 Recommender Systems

Recommender systems are responsible for suggesting items that might be a good fit for a particular user, be it movies, books, games, or others. The most used recommender system works by collaborative filtering, where each user records theirs reaction to a given item and then future users are recommended items that had a large amount of good reactions, in hopes of eliciting the same reaction [28]. Adding to this, some systems compute a similarity between users and through that can further filter the results by only matching good reactions to items from the subset of users that are more similar to the current user [29]. There are also recommender systems based on content-based filtering that try to recommend new items to the user based on previous items that the user liked and the similarities, in content, of the new item to the previously liked items [29]. Figure 2.3 shows the idea behind collaborative filtering and content-based filtering.

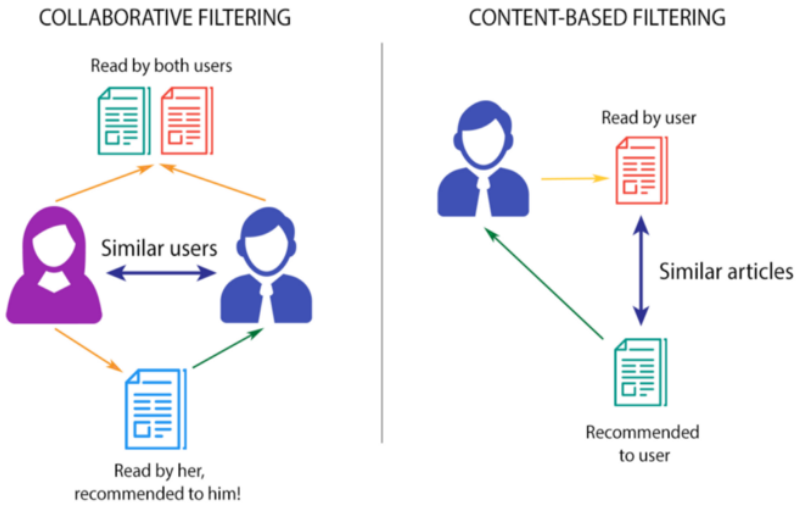


Figure 2.3: Exemplification of collaborative filtering and content-based filtering [30]

Recommender systems are used to recommend products on Amazon, or shows and movies on Netflix [31]. Regarding games, several approaches could be attempted regarding recommender systems. For example, given a procedurally generated level, players could rate the level and then future players would be recommended levels according to their similarity to previous players. Another example would be an arena based game where players were tasked with defeating different formations of opponents that the game decided. The player's enjoyment of the arena could be determined and arena formations recommended to future players based on a recommendation system method.

2.1.5 Summary

In this section of the state of the art, machine learning techniques were researched for possible inclusion on the solution. The techniques were reinforcement learning, transfer learning, imitation learning, k-NN, and recommender systems. Reinforcement learning consists of an agent maximizing a total cumulative reward in order to find a mapping of best action given a current state. Transfer learning is the idea of transferring knowledge between tasks that have similarities, applications in RL were analyzed in section 2.1.1.1 and its possible advantages also listed. Imitation learning consists of adapting the way that humans learn by mimicking was described, the main advantages of the technique and the overall workflow of the method was also seen. The k-NN algorithm classifies a new sample point by using the a previously classified set of samples. For that, the k nearest neighbors, through some distance measure, are computed and then majority voting is done on the classification the new sample should have. Finally, recommender systems try to recommend items to users by using previous users' ratings and the similarity of the current user to other users. The idea being suggesting items that users similar to the current one liked, in hopes of the current user also liking them.

As for the importance of said techniques: reinforcement learning could be used to train an agent to design interesting levels, interesting enemies, tweak game mechanics among others. Imitation learning could be used to train an agent to play the game optimally, or to train enemies to improve their success over the player. k-NN could be used to profile players so that future players could be profiled, having the player profiled can lead to knowing which game parameters work best for that player. Finally, recommender systems could also be used to profile players and then give them a suggestion that fits their profile. This could be applied to level designs, favorite enemy types, favorite gameplay content, and many others.

2.2 Game Adaptivity

The idea behind Game Adaptivity is to try to make a game more appealing to a particular player by taking into consideration their interactions with the game and then changing some aspect of the game to better fit the player [32]. To adapt any kind of content to the player, a model of the player is built by observing the player's actions and using that model, the content can be adapted to optimize a chosen metric: challenge (Dynamic Difficulty Adjustment), engagement, fun, among others [33].

At an architectural level, game adaptivity works by having a model of the player predicting the player experience based on the current game state. By feeding the player model and the predicted experience into an adaptation and generation engine, personalized elements of the game are output that should better fit the individual player. This architecture can be seen in Figure 2.4.

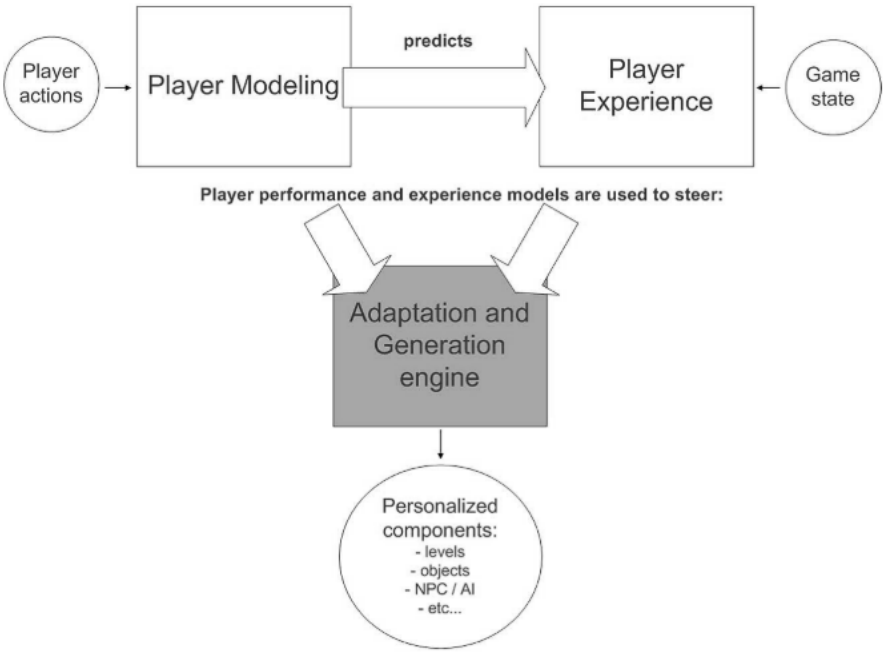


Figure 2.4: Game adaptivity architectural principles overview [1]

Now that a top-level view of Game Adaptivity has been performed, the next sections will focus on specific elements of the field. Section 2.2.1 will focus on listing and describing the types of game content that can be adapted as well as examples of such adaptations in the academic and commercial worlds. Section 2.2.2 describes the difference between offline and online adaptivity, followed by section 2.2.3 which performs a review of Player Modeling techniques and what criteria can be used for adapting a game. Finally, section 2.2.4 provides a summary of the Game Adaptivity section of the state of the art review.

2.2.1 Adaptable Content

Adaptable content in a video game can be divided into categories as follows [1]:

- **Game Worlds** - adapt game world or objects to better fit the player.
- **Mechanics** - adapt gameplay mechanics to better fit the player.
- **AI / Non-player character (NPC)** - adapt AI to better fit the player, most traditional approach to adaptivity.
- **Narratives** - adapt the game's narrative to better fit the player.
- **Scenario / Quests** - adapt the flow of actions and events to better fit the player.

Sections 2.2.1.1, 2.2.1.2 and 2.2.1.3 elaborate upon adaptation of the game world, the mechanics and the AI respectively. Adaptation of the narrative or of the scenario and quests of the game is not in the scope of the work.

2.2.1.1 Game Worlds

Adapting game worlds is concerned with how best to create, collate or modify a game world to better fit the player. Based on the player's model, a game can use procedural generation to modify or create a new world using the model as parameters of the generation process. There is also the concept of Designer-created Spaces which are spaces created for the game by its designers and are mostly static [34], one could use the player's model to link these Designer-created Spaces in a way that improves the experience.

The most well-known commercial game that employs this kind of adaptation is Valve's Left 4 Dead 2 which had a cemetery level that changed its layout to be more or less complex according to the group's performance [3].

On an academic level, research has been done on improving the fun of a racing game by generating tracks that tried to take into account how fun the tracks would be perceived by the human players [35]. In a platformer game, fun was maximized and frustration and challenge balanced to improve the game experience by generating levels taking into account player style and predicted player experience that the level would provide [36]. Both of these examples of game world adaptation concerned maximizing the metric of fun. Another metric that can be optimized

is difficulty, one such example generated 2D platformer levels by considering the player's skill level and estimating the difficulty of generated level segments so that they would roughly match the skill level of the player [37].

2.2.1.2 Mechanics

Game mechanics include the rules of the game and the actions that a player may take. Regarding the rules, an adaptation of the mechanics could fine tune damage formulas, health values, knock-back distances among others to alter the game experience. As for the actions available to the player, an adaptation could increase the jump height, the tolerance of the aiming, the speed of the player among others.

An example of a commercial game that adapted game mechanics to fit the player is Max Payne, which adjusted the aim assist of the shooting mechanic according to how good the accuracy of the player was [1]. Further along the series, Max Payne 3, also featured adaptive elements such as restarting checkpoints with more health restoring items and ensuring the player had ammunition even if they didn't when dying [38].

On an academic level, an educational game to teach microbiology has seen its gameplay elements adapted, such as disabling the countdown clock, according to the type of player, which was determined through a questionnaire [39]. Another example was a first person shooter game was adapted in several ways, one of them being a mechanical change that adjusted the blast radius of weapons so that players whose skill was considered to be at an amateur level could not inflict damage on themselves as easily [40].

2.2.1.3 AI / NPC

Adapting the AI that governs the decisions of the game's opponents or adapting the AI of the NPCs involves tweaking them so that they make different decisions depending on the player they are dealing with. Traditionally, this adaptation is the most researched area [1], and it can vary greatly depending on the objectives of the adaptation. Enemies can become more challenging by adapting their AI to counter specific tactics the player is employing, or on a more educational context, the NPCs that serve as tutors for the players can become more effective by altering their behavior to match the needs and preferences of the player, a particularly important consideration when dealing with serious games [41].

Commercial games that have employed techniques of AI adaptation include Grand Theft Auto 5 [42] and Pro Evolution Soccer 08. In Grand Theft Auto 5, the racing mode of the game featured a technique by which the leading cars would be slowed down so that the player could more easily catch up to the lead position, this is informally called "rubber banding" and is present in many Racing games such as the Mario Kart series [43]. As for Pro Evolution Soccer 08, it implemented an adaptive AI called "Teamvision" which changed the opposing team's tactics to better counter the player [40].

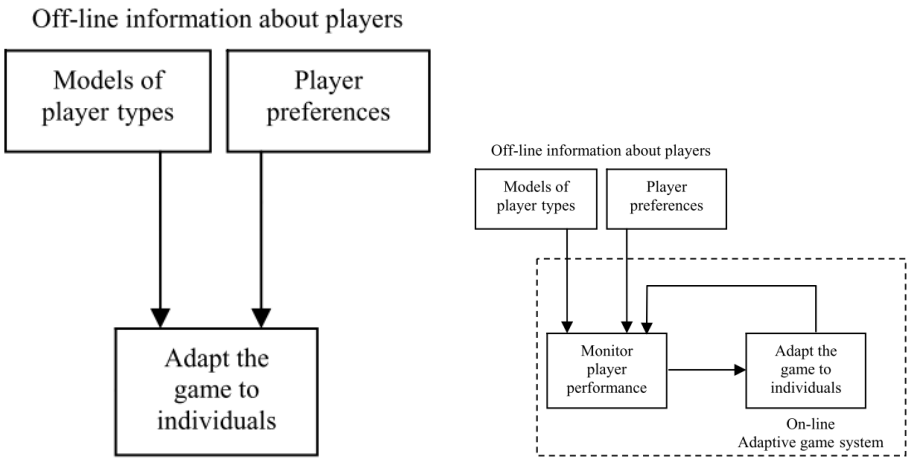
On an academic level, the AI of a Real Time Strategy game was adapted to the opponent by using opponent modeling techniques [44]. Another work used Unreal Tournament [45] to implement incremental case-based approaches to modeling and predicting to adapt an NPC's decisions based on observations of the player [46]. Additionally, another work adapted its AI (tactics) when dealing with a confrontation between two opposing teams in a Role-Playing Game setting by using reinforcement learning techniques [47].

2.2.2 Types of Adaptivity

When implementing game adaptivity in a game there are two options to choose from at an architectural level: performing it offline or performing it online [1]. This basically means that the adaptation runs entirely before the player experiences the content (offline) or that the content is continuously adapted taking into consideration the latest player actions (online). These types of adaptivity can be applied to any of the adaptable content seen in section 2.2.1.

Offline adaptivity generally involves using content generation techniques to create tailored content based on a previously obtained model of the player, which means that offline adaptivity is mainly a content generation challenge. Content can be generated as the player transitions between levels but in a given level the content is no longer changed in offline adaptivity, the adaptation runs entirely before gameplay. This type of adaptivity has been mostly been used in the procedural generation of game worlds [1].

Opposed to offline adaptivity, online adaptivity attempts to tailor the content to the player as they play the game. A framework for generically implementing online adaptivity was proposed by Charles *et al.* [48] which consists of modeling the player to know its skill level and habits, then during gameplay analyzing the player performance and in tandem with the player model identify possible adaptations. Adaptations that are performed are measured for their effectiveness, using a chosen metric, which can lead to an update of the adaptation or of the player model. Since the player model and performance are updated while playing, this type of adaptivity can respond to playing style changes of the player, for example a sneaky player changing to a more hands-on approach to combat. Possible architectures behind offline and online adaptivity are shown in Figure 2.5.



(a) Offline adaptivity (b) Online adaptivity

Figure 2.5: Offline and Online adaptivity systems, adapted from Charles *et al.* [48]

2.2.3 Player Modeling

As mentioned in the introduction of section 2.2, a player model is used to predict the player experience so that it can be used as input for the adaptation engine. To effectively model the player, this section overviews player models that are found in academia and the types of metrics they analyze.

Bartle proposed a simple taxonomy of players dividing them into: achievers, explorers, socializers and killers. Achievers play games with a goal oriented perspective, they are always trying to achieve some goal they set out to complete. Explorers are interested in exploring the game world and the rules that govern the game (e.g. mechanics, physics). Socializers play games for the social aspect and interaction, for example by role-playing or by cooperation / competition. Finally, killers play the game with the intent of imposing on other users, usually by getting involved in player killing activities [49].

The GameFlow model analyzes player enjoyment by dividing it into eight elements: concentration, challenge, skills, control, clear goals, feedback, immersion, and social interaction. **Concentration** is how good a game is at keeping the player's attention, **challenge** is how much a player feels like their skills are being challenged by the game, **skill** concerns how good a game is at teaching the player and how adequate its learning curve is, **control** captures whether players feel like they are in control of their actions in the game, **clear goals** means that the player knows at all times what their objective is, **feedback** analyzes the appropriateness of the game's feedback as well as its timeliness, **immersion** concerns the involvement that the players feel when playing the game, **social interaction** is how much of a social aspect the game features such as player cooperation and interaction. When evaluating this model it was deemed adequate for analyzing player enjoyment but the different criteria for each element were found to work better in particular genres [50].

The Five Factor Model (or Big Five) is a psychological model of personality types which analyzes personality as a combination of five factors: openness, conscientiousness, extraversion, agreeableness and neuroticism. **Openness** is associated with originality, creativity, and exploration behaviors, **conscientiousness** is associated with punctuality, self-discipline, and organization, **extraversion** is associated with enthusiasm, energy, and sociability, **agreeableness** is associated with altruism, modesty, and compassion, **neuroticism** is associated with nervousness, poor coping capacity, and emotional instability [51]. A relationship between player personality and game enjoyment and gameplay duration was found when adapting the difficulty of a First-person shooter by choosing from four difficulty adaptations. These difficulty adaptations were chosen by using the Five Factor Model. Enjoyment and gameplay duration was found to be positively correlated to certain personality types. A more complex adaptive system based on this personality model was deemed plausible by using the results obtained [52].

The listed models show that very different approaches exist for player modeling, Bartle's approach is a very simplified way of categorizing players into four categories which have no overlap, the GameFlow model analyzes the game instead of the player to predict the enjoyment that a player would have from playing that game, and the Five Factor Model quantifies personality by dividing it into five factors and measuring how strong each is. These models are not compatible with each other and offer different possibilities for the criteria that can be maximized by the adaptation process, for example the GameFlow model is meant for analyzing player enjoyment and nothing else. In contrast, the Five Factor Model can be applied to different criteria depending on the implementation. For example Nagle *et al.* [52] used the model for maximizing enjoyment and gameplay duration in a First-person shooter but deemed it possible for other adaptations.

2.2.4 Summary

In this section of the state of the art, game adaptivity was researched since it is the main contribution of this work. An overview of what game adaptivity consists of was provided and the types of adaptable content in games was listed in section 2.2.1 A short description of game world adaptation, mechanics adaptation and AI / NPC adaptation was provided, as well as examples of commercial games and academic research that implemented each of these types of content adaptation. Section 2.2.2 describe offline and online adaptivity. Offline adaptivity consists of generating content before the gameplay while online adaptivity consists of running the adaptation while the play session is running. An example architecture of offline and online adaptivity was provided. Finally, player modeling was analyzed in section 2.2.3 describing some of the main works in this field. Models vary greatly and were found to be largely incompatible with one another, the criteria that each model could possibly be used for also changes depending on the model. Choosing the criteria to optimize in this work will have a direct impact in the available models to use.

In conclusion, for the design of the solution, the main points to consider are: what type of content we want to adapt, how that content should be adapted, and what criteria should be used to measure whether there was a significant change.

2.3 Game Adaptivity using Machine Learning

After the analysis of the current ML techniques in section 2.1 and having performed an overview of game adaptivity in section 2.2, the question that stands is whether these two fields can be used together to produce interesting results. This section explores the latest works found that used ML techniques successfully in adapting a game. The list is presented in chronological order, sorted from oldest to newest.

Reinforcement learning has been successfully used in dynamic difficulty adjustment, a type of game adaptivity, in the context of adapting the AI of the opponent character in a Fighter game so that the opponent would match the player's skill level [53].

MimicA, a framework developed for Unity adapts companion character's AI to that of the player based on the idea of imitating the player with the goal of raising fun and immersion [54]. Very similar to the idea behind imitation learning, presented in section 2.1.2.

Middle-earth: Shadow of War [55], a commercial game released in 2017 featured the "Nemesis System" which constructed personalized narratives with certain enemies and followers of the player. The player dying at the hands of the enemy altered the personal narrative of the enemy that defeated the player by giving it personalized upgrades while the way in which the player obtained an orc follower shaped that orc's features. This created a personalized narrative bubble for each relevant enemy/follower and an overall different narrative for each player [56].

Behavioral Repertoire Imitation Learning (BRIL), a method developed to perform build-order planning in the game Star-Craft II that augments the process of imitation learning by attempting to learn multiple behaviors instead of a single "average" behavior from the available experts. This resulted in a single neural network capable of exhibiting several behaviors depending on the state description of the Start-Craft II game sessions [57]. Tested only with bots, this adaptation of the AI could be implemented in the context of dynamic difficulty adjustment.

Using Unity, a Role-playing game was developed which had a game story writer AI whose objective was to craft unique stories by taking into consideration the actions of the player as well as that of the NPCs. This approach uses basic reinforcement learning to control NPC's behaviors towards each other. While the ML approach is very basic, this adaptation is very interesting because it adapts narrative, scenario and NPC behavior at the same time with the goal of improving the gameplay experience through story personalization instead of the more classical difficulty adjustment approach [58].

2.4 Summary

Concluding the state of the art review, machine learning was researched in section 2.1 and current techniques were analyzed, reinforcement learning in section 2.1.1 and imitation learning in section 2.1.2, followed by sections 2.1.3 and 2.1.4 about k-NN and recommender systems respectively. These techniques were researched since they are candidates to use in the proposed solution of this work.

Game adaptivity was reviewed in section 2.2 describing what content could be adapted in section 2.2.1, the types of adaptivity in section 2.2.2 and section 2.2.3 on player modeling. Content that can be adapted in games is categorized into: game worlds, mechanics, AI / NPC, narrative, and scenario / quests. Types of adaptivity consist of offline and online adaptivity, the main difference being whether adaptation is done while the play session is occurring. Player modeling consists of techniques meant to categorize the player, a useful input for the game adaptation process. The main conclusion is that the criteria to optimize when adapting a game influences the available models to use, and that the models are usually not compatible with each other.

The final section of the state of the art review explored related works that had successfully implemented game adaptivity using machine learning techniques. The techniques seen in section 2.1, Reinforcement learning and imitation learning, were both used successfully in adapting a game.

Chapter 3

Proposed Solution for the Adaptive Game

Having researched the state of the art on adaptive games, machine learning and the intersection of the two fields, the planning and development phase could start. This chapter focuses on the concept of the proposed solution, along with decisions that were made in order to bring the concept into the implementation domain. Afterwards, the implementation is detailed with its architecture, tools and technologies, and the specifics of the development itself.

In terms of structure, the chapter begins with the description of the concept in section 3.1. Then, the decisions made to move from concept to implementation are detailed in section 3.2. The following sections 3.3 and 3.4 describe the architecture and technologies used for the implementation. Finally, sections 3.5 and 3.6 relate the development process and specifics of the implementation.

3.1 The Concept

As seen in section 2.2, an adaptive game tries to make a regular game more appealing to a given player by taking into account the type of player that is currently playing. The intent of the game developed for this work is to assess whether a chosen Machine Learning technique is suitable for creating an adaptive game experience that improves upon the non-adaptive game.

Conceptually, the game should be developed in a way that makes it possible to alter a finite set of variables that govern the game experience, thus making the game parameterizable. Changing these variables impacts the game in different ways, one of the variables might decide the total health points the player has, another might control the damage output of the player and so on. The variables are not restricted to simple player/enemy stats modifications either, they might be procedural generation parameters, enemy AI related parameters, or even plot related parameters such as sequence of events. This means that a parameterizable game is capable of minor modifications

such as game balancing (e.g. health / damage / speed), but also major ones such as level design, enemy AI (e.g. aggressiveness / actions available) and plot flow (e.g. which events happen / order of events / characters that appear). At the conceptual level, a parameterizable game is potentially capable of adapting any of the content discussed in section 2.2.1.

For the tuning of the game's parameters, some form of algorithm or process must be present so that the game can be said to be adaptive. A player model or player preferences can be used to steer the adaptation process, as seen in section 2.2.3. These can be obtained beforehand or inferred during gameplay, and sometimes systems combine both approaches and update a previously obtained model with new observations. The idea is that an algorithm tunes the parameters of the parameterizable game in respect to a given goal and a given input of player model or preferences. This means that the parameters to tune depend on the current goal-model pairing or goal-preferences pairing. For example, if the goal is to modify the level design in a way that makes it less prone to players getting lost, the algorithm would tune the procedural generation parameters and not tune any other parameter. Previous players' performances or current player performance can also be used to further tune the parameters. Going back to the example of changing the procedural generation so that less players get lost, one could use the previous performances and the parameters at the time, to know which sets of parameters lead to bad or good results. This can be applied to any given goal, and in the case of Machine Learning algorithms can be used for inferring good sets of parameters with varying degrees of complexity depending on the chosen algorithm.

The overall architecture for the concept of adapting a game has an adaption algorithm at its root, which gets input from the player model, player preferences and/or player performance, that then tunes parameters of a parameterizable game so that the overall output becomes an adaptive game. Optionally, the adaptive game can connect back to the inputs of the adaption algorithm so that the adaption is constantly being updated, thus making the system a loop. This can be seen in Figure 3.1.

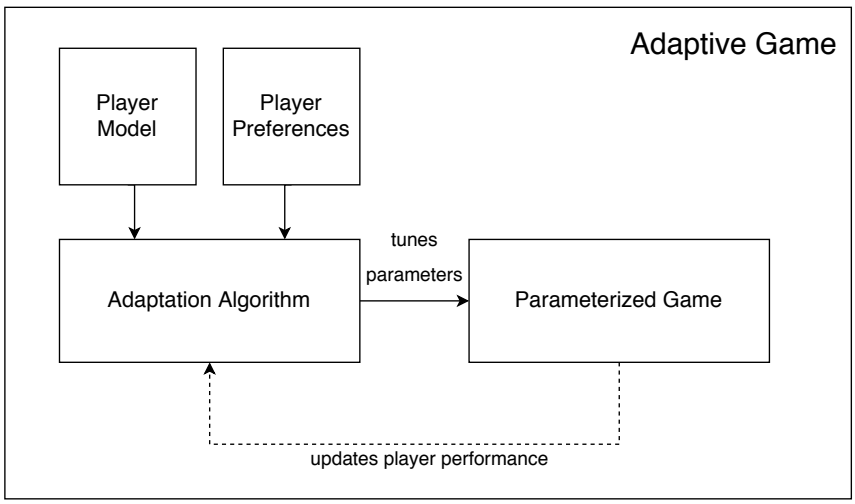


Figure 3.1: Conceptual architecture of an adaptive game

Knowing the overall picture of an adaptive game at its conceptual level, steps can be taken to take it to a more specific solution that can be implemented. While the conceptual level can be implemented through a set of abstraction layers that turn the adaptation engine into a framework, thus making it possible to adapt any game in respects to any given goal (e.g. level design, difficulty, plot) that is not the goal of this thesis. The goal is analyzing whether a certain machine learning technique can create an adaptive game that results in improved game experience when compared with the non-adaptive game. Since a framework is not the end goal, certain elements have to be decided beforehand as to reduce the scope of the implementation, namely genre and what type of content to adapt. Genre agnostic or type of content agnostic adaptation engines might be a possibility but are outside the scope of this work.

3.2 From Concept to Implementation

As described in section 3.1, while the overall conceptualization of an adaptive game could be implemented as a very flexible adaptation framework, the idea is to explore whether a Machine Learning approach could lead to an adaptive game with improved player experience. To better focus on this task, a well defined game genre and type of content to adapt had to be decided.

Genre choice did not take long since the idea of developing a game similar to the dungeon aspect of the The Legend of Zelda [59] series was something I wanted to do. Still, whether this genre was a good choice for the desired adaptation method was also considered. Due to the type of content to adapt not being decided yet, a genre where it would be possible to easily adapt most, if not all, of the types of content would be a fitting choice.

Having chosen the genre, the choice of the type of content to adapt remained. For that choice, the possibilities of each type where considered. For example, in the domain of Game World adaptation, one could change the placement of items, the enemy placement or even the level boundaries and layout to create a completely different experience. For that the game would have to include these features as to allow tweaking them. As for the Mechanics, the type of interactable objects available could be changed, for example puzzles with pushing blocks could appear to some players but to others it would be puzzles to reflect lasers with mirrors. The damage formulas or game entity attributes could be tweaked as well. All of these are common in the chosen genre. In regards to the AI, the obvious solution would be to alter its behavior so that it matched the player's performance, a typical scenario of dynamic difficulty adjustment. There could also be a companion NPC which could have its behavior tweaked to match the player's style. Finally, Narrative and Scenario adaptation are not considered for the adaptation as they would need further research to create a compelling scenario and narrative. However, if they were considered, one could have plot variations for different player profiles in the case of Narrative adaptation, and for Scenario adaptation, the type or order of quests received could be adapted to the player. All of the adaptation options are summarized in Table 3.1. For a refresher on adaptable content please see section 2.2.1. With these considerations, it seems pursuing any of the adaptable content types is possible with the chosen genre.

Adaptable Content	**Examples of adaptation for the chosen genre**
Game World	Powerup placement / Item placement / Enemy placement / Level design
Mechanics	Types of puzzles available / Damage formulas / Player and NPC attributes tweaking
AI / NPC	Enemy behavior
Narrative	Plot variations
Scenario	Sequence of events / Variations of the main quest

Table 3.1: Adaptable content examples for the chosen genre

Having considered the different adaptation possibilities, one must be chosen for implementation. As mentioned in section 2.2.1, narrative or scenario adaptation is out of the scope of this work so those were already excluded. This leaves game world, mechanics and AI as possible adaptation avenues. AI adaptation was the first to be discarded, since most options would lead to dynamic difficulty adjustment which is the most common field of game adaptivity. It would be a perfectly fitting adaptation with the chosen genre but due to it being the most explored avenue, I decided to try one of the alternatives. Mechanics also had to be discarded because most options available for the planned game were also dynamic difficulty adjustment. The scope of the game would have to increase in order to have a number of mechanics that could be turned off or combined in a way that created a new game experience, failing that, only formulas and statistics could be adapted which mostly leads to the aforementioned dynamic difficulty adjustment. Game world adaptation was the only element left, and since the dungeon was to be procedurally generated, it would be possible to tune the generation parameters. Thus, game world adaptation was the chosen content to adapt. While multiple types of content could be adapted, choosing only one helps reduce the scope but, more positively, isolates the adaptation so that conclusions can be drawn as to whether the method lead to a better player experience in regards to game world adaptation.

This design and concept considerations leads to the prototype, which uses a 2D Top-Down Action game, akin to the The Legend of Zelda series, with tuning of the procedural generation parameters in order to create an adaptive game. Details of the parameters, the algorithm that tunes them, as well as the criteria used for adaptation will come later in section 3.6. For now, a look at the architecture and technologies of the implementation will follow in sections 3.3 and 3.4.

3.3 Architecture

The architecture of the solution is composed of three main components: the parameterizable game, the adaptation process (ML algorithm) and the user data. The parameterizable game consists of a 2D Top-Down Action game, where the procedural generation of the dungeon floor is parameterized in a way that allows the adaptation process to tweak it. The adaptation process consists of a k-NN algorithm that uses the current player's floor performance and the previous players' stored performances to tweak the procedural generation of the next floor. Finally, the user data is output by the game during the play session, which can then be combined from various players in order to

create a set of previous play session data to be used as input for the adaptation process. Figure 3.2 illustrates this.

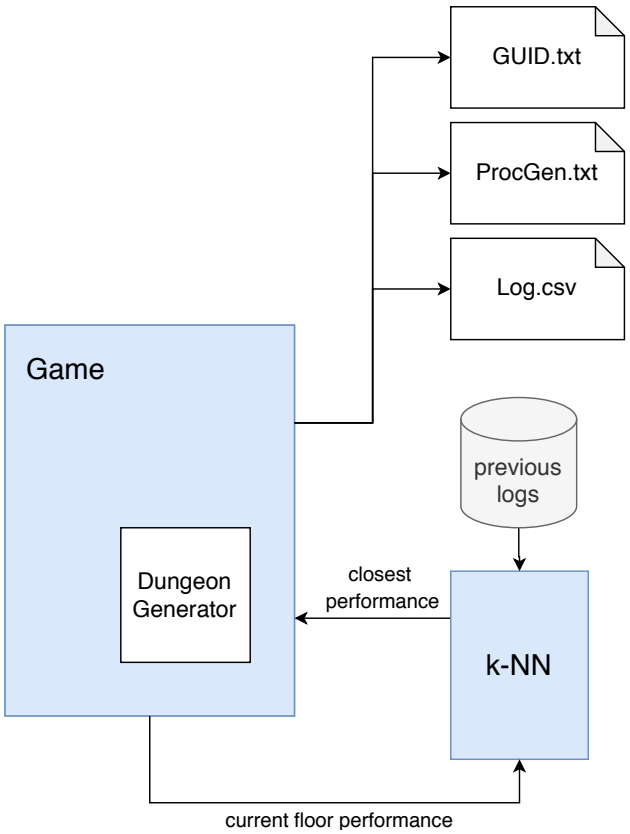


Figure 3.2: Architecture of the proposed solution

Each of the components is detailed in further sections. For the game, section 3.5 gives an overall description of its functionalities and how it plays. For the adaptation process, section 3.6 details the process of choosing an algorithm compatible with the developed game. For the user data, section 3.5.7 talks about the reasoning for logging the user performance and section 4.2 details what is gathered in an exhaustive manner.

3.4 Tools and Technologies

Having the architecture and game specifics decided, choosing the technology to implement them was possible. During the research phase, Unity[1] and Unreal Engine[2] were considered as I have experience with both and they are very versatile game engines. Unreal Engine was discarded in favor of Unity due to better Machine Learning plugins and ease of use when dealing with external tools or software. Unity has a plugin, named ML Agents, dedicated to Machine Learning tasks in-engine. Even if the implementation were to not use it, the option of it being available was more

[1] https://unity.com/
[2] https://www.unrealengine.com/en-US/

important than what Unreal Engine could offer. Unity version 2019.3.13f1 was used without any additional plugins.

Python[3] was also used for processing user test data as needed, being a tool regularly used for this purpose, due to its ease of use and variety of packages that extend the basic features. Most of the graphs / charts were done using Google Sheets[4].

The game was distributed as a binary file built in Unity, so choosing a host was needed in order to share the game for user testing. Itch.io[5] was chosen since a cursory search revealed that it is regarded as the best place to host independent games. It allows hosting of the game free of charge, with native support for Operating System (OS) specific versions of the game which helped distribute the game for Windows and Linux. It also allows the uploader of the game to write a HTML page presenting the game which allowed the usage of said page as a landing page for the user test. Other game platforms exist but were not considered since they are either difficulty to gain access to, or have associated fees.

3.5 The Non-Adaptive Game

This section describes the implemented non-adaptive game, which serves as basis for the adaptive version. To better understand how the game works, its gameplay is explained in subsection 3.5.1. Subsection 3.5.2 talks about the different screens the players see and describes the elements included. The following subsections talk about specifics of the characters (section 3.5.3), combat (section 3.5.4, and puzzles (section 3.5.5. The procedural generation of the game levels and its parameters are described in subsection 3.5.6, ending with an overview of the logged user data in subsection 3.5.7. At the end of this section, the reader will have the information needed to better understand the adaptation process that follows in section 3.6.

3.5.1 The Gameplay

The game consists of a dungeon with infinite floors that are generated as the player reaches them. The floors are composed of same sized rooms and can have a varying number of rooms. The rooms are connected through doors that are already open or have a condition to be opened. Dead ends exist, which makes the floor layout work in the same way a maze does. The goal of the player is to make it from the start room to the exit room so that they can proceed to the next floor. Figure 3.3 shows an example of the game's floor. The first five floors have different procedural generation parameter ranges, and beyond that point, they always follow the same ranges. Besides the start and exit rooms, rooms can contain combats, puzzles or just be connecting corridors. The player has a limited amount of health points which regenerate between floors. Depleting the health points results in a game over state, which takes the player back to the main menu. Saving or continuing a previous play session is not possible, every attempt at the game is a self contained play session,

[3] https://www.python.org/
[4] https://www.google.com/sheets/about/
[5] https://itch.io/

which ends up helping evaluating the user's game experience since there is no way for the user to interrupt it. By tackling the floors' challenges, the player is tasked with reaching the deepest floor they can.

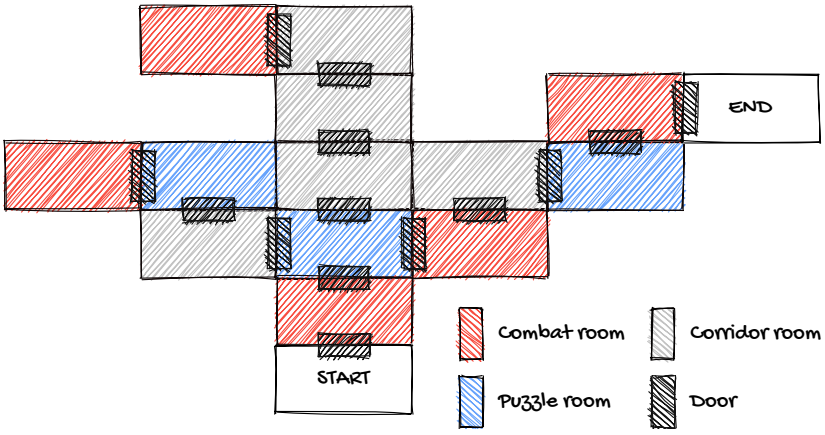


Figure 3.3: Example of the game's floor

From a scene flow perspective, the game starts with the main menu. From there, the player can start a new game, edit game options or exit the game. By starting a new game, the game generates a new floor for the player, which the player is then tasked to complete. Floors are generated infinitely until the player dies or decides to quit. If the player dies, the game over screen is displayed and the player is taken back to the main menu. This flow of game scenes can be seen in Figure 3.4.

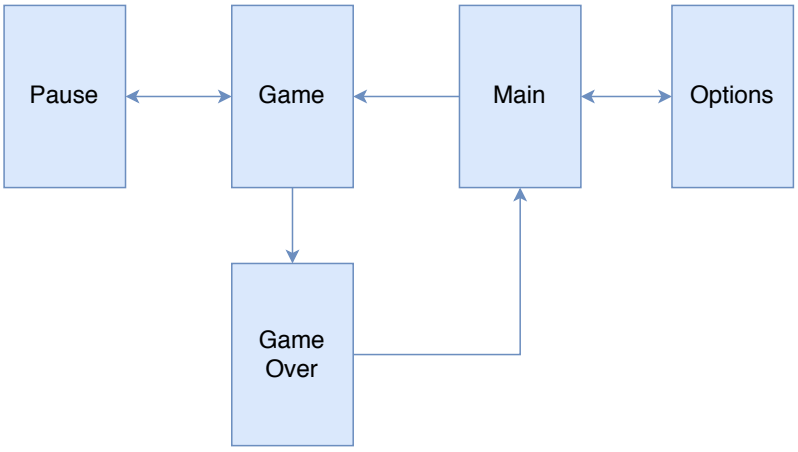


Figure 3.4: Flow of game scenes

3.5.2 User Interface

The game's user interface will be presented on this section. The main menu allows starting the game as well as accessing the options menu. The options menu allows choosing the resolution of

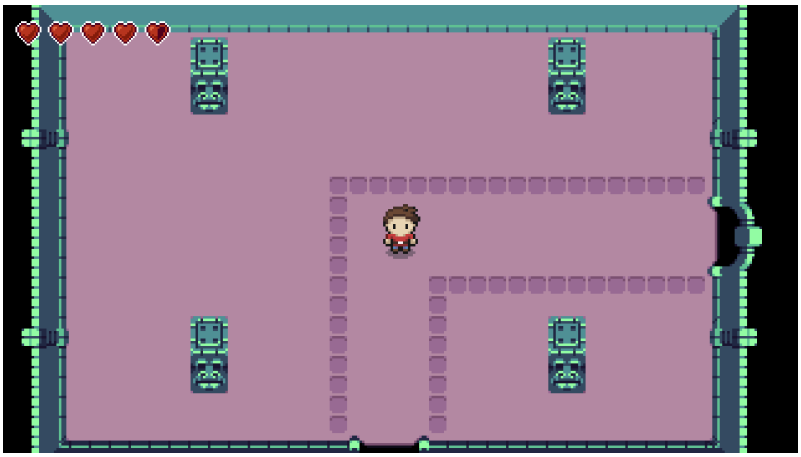
the game as well as whether the game should run in fullscreen mode. Both menus can be seen in Figure 3.5.

(a)

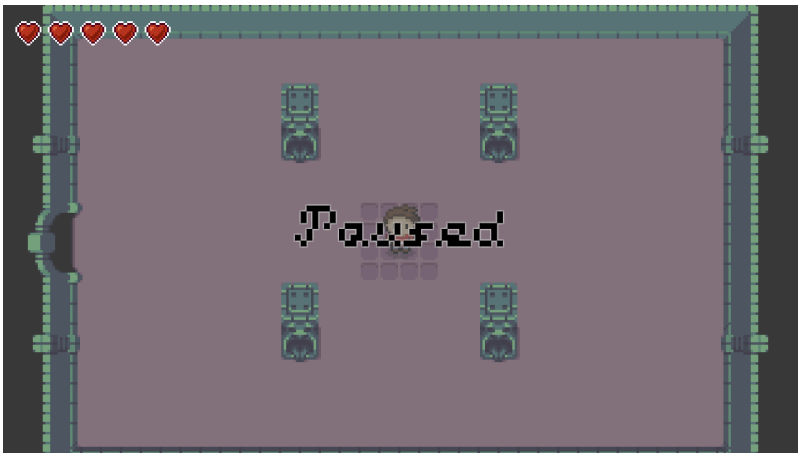
(b)

Figure 3.5: The game's main menu (a) and options menu (b)

When playing, the health points are shown as heart icons on the upper left of the screen. As the player loses health points, they display as empty heart icons. The player can also pause the game at any time during gameplay. An example screenshot of playing the game and pausing it can be seen in Figure 3.6.



(a)



(b)

Figure 3.6: The gameplay (a) and the pause screen (b)

Finally, the game over screen appears whenever the player's health points are depleted, returning the player to the main menu. The game over screen can be seen in Figure 3.7.

Figure 3.7: The game over screen

3.5.3 Player and Enemies

The game features the player character, along with two enemy types: the Log enemy, and the Skeleton enemy. Each character or enemy, has a max health point value, an attack damage value, and a speed of movement value. The player character can move in an 8-directional manner and attack with a sword slash in the four cardinal directions. This can be seen in Figure 3.8. The player character has the fastest movement speed in order to make game fairer to the player.

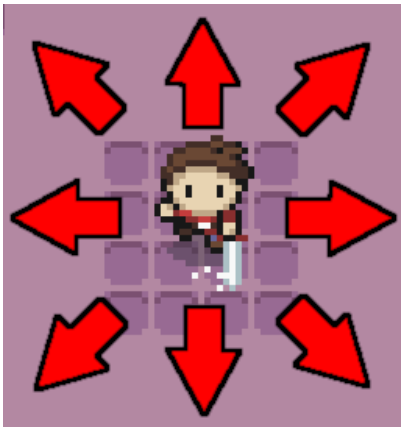


Figure 3.8: Player attack and possible movement directions

As for the enemies, the log enemy can move in the same directions that the player can albeit slower. Initially, the log enemy is sleeping and when the player gets too close, it wakes up and begins following the player in order to attack. Attacks from the log enemy are dealt as contact damage, meaning the log only has to touch the player to attack, a cooldown prevents the log from attacking every frame when touching the player. If the player gets far enough from the log enemy, it goes back to a sleeping state waiting for the player to come near again. Figure 3.9 shows the log enemy sleeping and the radius in which it wakes up and follows the player.

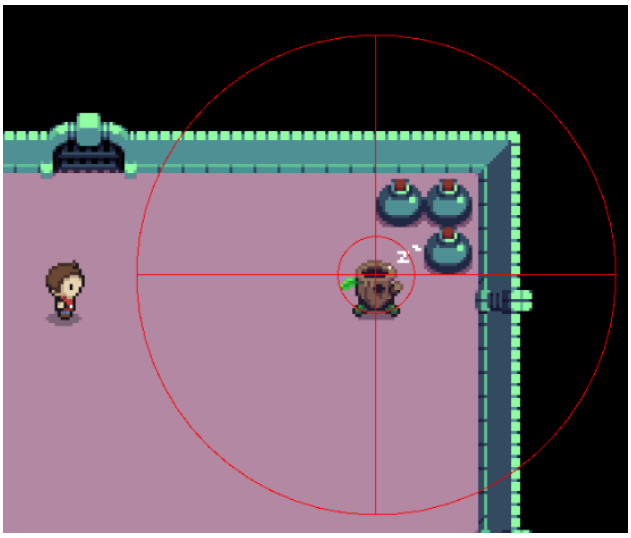


Figure 3.9: Log enemy sleeping, the outer radius marks the point where it wakes up and follows the player

The skeleton enemy behaves completely different from the log enemy. It is a ranged attacker that tries to hit the player with its arrows. When the skeleton notices the player, it gets closer in order to shoot its bow at the player. As long as the player is in range, the skeleton acts as a turret, standing in place firing arrows at the player. If the player gets far enough, the skeleton gives chase in order to continue shooting arrows. The bow has a cooldown so that the enemy has to wait between arrow shots in order to make the game fairer. Figure 3.10 shows the skeleton's ranges.

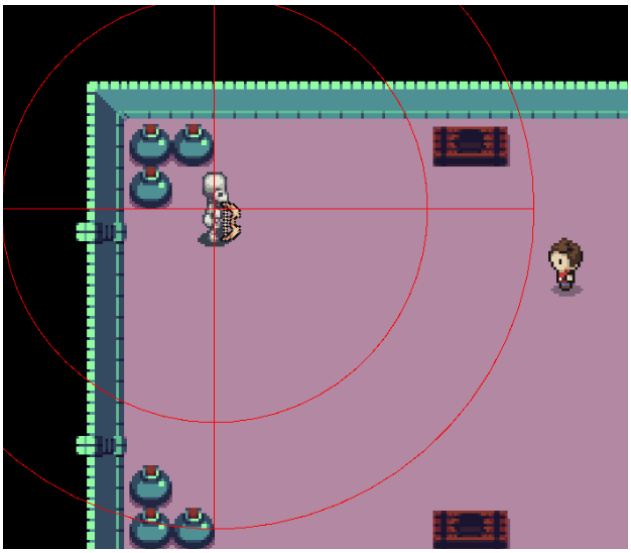


Figure 3.10: Skeleton enemy waiting. When the player reaches the outer radius, the skeleton closes the distance until the inner radius where it starts firing

3.5.4 Combat

Combat segments happen whenever the player stumbles upon a new combat room. The room's doors close and the player has to defeat the enemies that appear in order to open the doors. Combat rooms that have been cleared by the player do not deploy new enemies, allowing safe passage. Combat rooms spawn combinations of log and skeleton enemies depending on their difficulty: easy, medium or hard. Each difficulty has a total amount of enemy points that can be used to fill a combat room. Whenever the player enters a previously unseen combat room, the game generates a combat by looking at the difficulty and populating the 4 spawn points with enemies that respect the maximum enemy points allowed. This generation consists of picking, at random, from a list that contains all the possible formations for each difficulty. No more than 4 enemies may be placed at the same time. Table 3.2 shows how many enemy points each difficulty allows, and table 3.3 shows the cost, in enemy points, of each enemy.

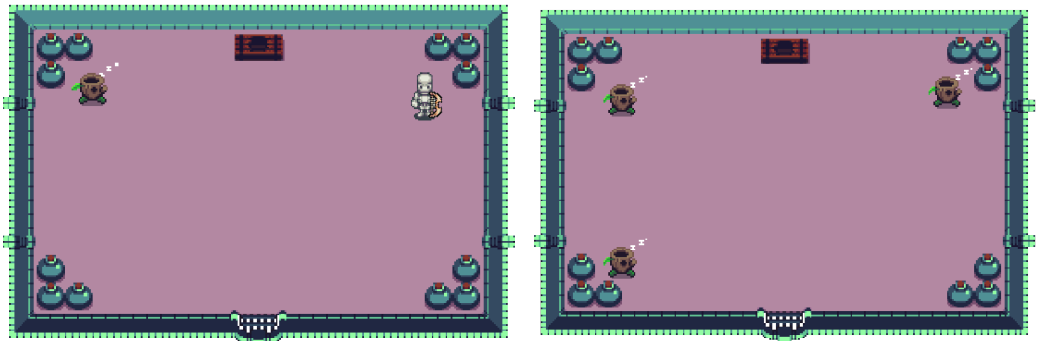
Difficulty	**Max Enemy Points**
Easy	2
Normal	3
Hard	5

Table 3.2: Max enemy points per difficulty

Enemy	**Enemy Point Cost**
Log	1
Skeleton	2

Table 3.3: Cost of each enemy in enemy points

To better illustrate the spawning mechanic, Figure 3.11 shows a comparison between two normal difficulty combat rooms featuring two different enemy formations that respect the maximum allowed enemy points for that difficulty. The formation picker always tries to maximize the used enemy points, since otherwise it could lead to higher difficulty rooms spawning formations of easier difficulties by allowing too many unspent enemy points.



(a) (b)

Figure 3.11: Example of two different normal combat room enemy formations

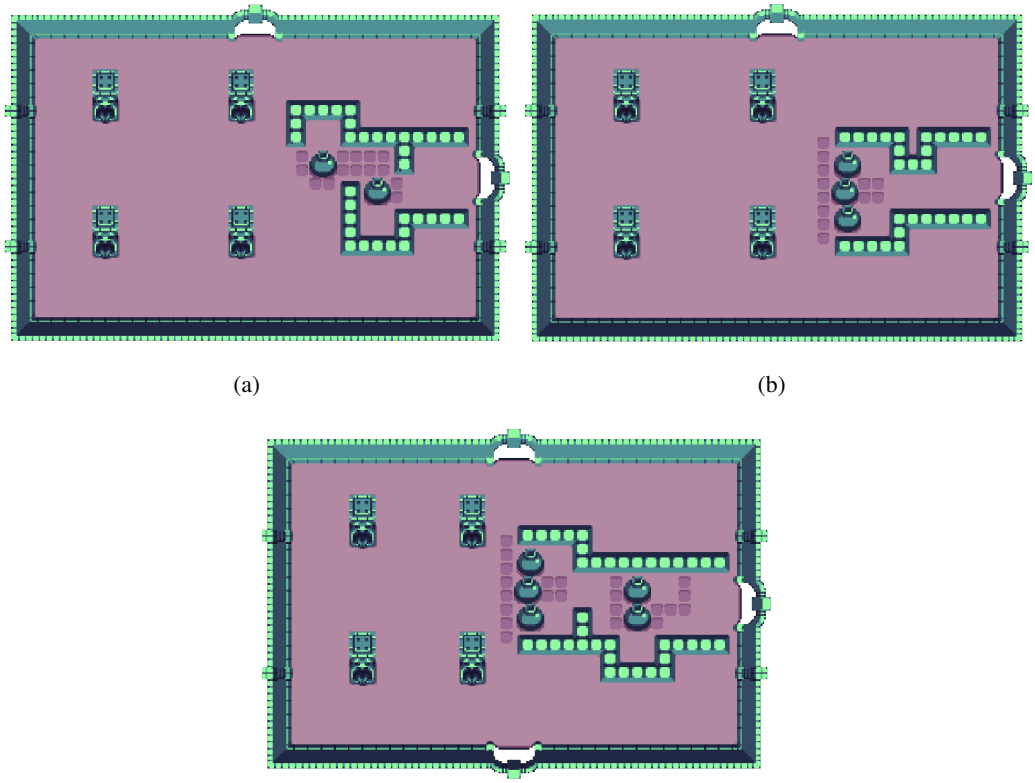
3.5.5 Puzzles

The player encounters puzzles whenever they stumble upon a puzzle room. The goal is to push the pots to open the way to the exit that they block. When the player manages to exit through the door blocked by the pots, the puzzle is marked as complete and the pots stay as the player left them. If the player leaves without solving the puzzle, the puzzle resets itself to the default configuration, allowing the player to restart in cases where the pots get stuck. Puzzles also feature the same difficulty levels as the combats do, but in the case of puzzles, the difficulty level is used to choose between pre-made puzzle layouts of the desired difficulty, of which there are two per difficulty. Table 3.4 shows how many pots the player has to clear out of the way per difficulty.

Difficulty	**Number of Pots**
Easy	2
Normal	3
Hard	5

Table 3.4: Number of pots to push per difficulty

Figure 3.12 shows puzzle layout examples for each difficulty.



(a) (b)

(c)

Figure 3.12: Example of an easy puzzle (a), normal puzzle (b) and hard puzzle (c)

3.5.6 Procedural Generation - Dungeon Generator Component

The Dungeon Generator component is responsible for generating floors whenever the player reaches a new floor. The procedural generation takes into account the current floor number up to the fifth floor, beyond the fifth floor the generation parameters' ranges remain the same. The process, step by step, goes as follows:

1. Randomize dungeon size (horizontal/vertical dimension).
2. Choose a random corner as the player's starting location.
3. Randomize shortest path length, meaning the minimum number of rooms the player has to traverse to reach the exit.
4. From the start position, random walk until the shortest path length is reached, place exit at end of path.
5. Find maximum number of dead ends that can be placed.
6. Randomize number of dead ends to place based on the maximum available.
7. Generate dead ends through random walking.

8. For each empty room, randomize whether that room is a combat, puzzle or corridor room.

9. Done!

Elaborating on some of the listed steps, dungeon size randomization has a probability of generating a dungeon with x horizontal cells and y vertical cells, according to the current floor number. Table 3.5 shows the probability of a certain dungeon size given the floor number.

Floor Number	**2x2**	**3x2**	**2x3**	**2x4**	**4x2**	**3x3**	**3x4**	**4x3**	**4x4**
1	10%	35%	35%	10%	10%	0%	0%	0%	0%
2, 3	0%	15%	10%	25%	25%	25%	0%	0%	0%
4, 5	0%	0%	0%	0%	0%	15%	35%	30%	20%
>5	0%	0%	0%	0%	0%	0%	35%	35%	30%

Table 3.5: Probability of each dungeon size per floor number

Having the dungeon size and a starting position, the length of the path from the starting room to the exit room is randomized. For this, the minimum and maximum path length is computed for the current dungeon size. The minimum is always 3, to allow a room between the start and exit rooms, and the maximum is always the dungeon area. Instead of doing a randomization of the length with this minimum and maximum, the current floor number modifies the range in order to make the deeper floors larger. Table 3.6 shows how each floor modifies the range, *diff* is computed as $maxLen - minLen$.

Floor Number	**Length Randomization Behavior**
1, 2, 3	Range is [min + 0.25diff, max - 0.25diff]
4, 5	Range is [min + 0.50diff, max - 0.25diff]
>5	Range is [min + 0.50diff, max]

Table 3.6: Range of shortest path length randomization per floor number

With a start position decided and a path length from that starting position to the exit also decided, the path can now be generated. For this, a random walk is performed from the start position until the desired length is reached. The random walk consists of:

1. From current cell compute adjacent cells.

2. Compute valid cells, that is, cells that have not previously been visited and are in-bounds.

3. Choose at random one of the valid cells and move there.

4. Stop if desired path length has been reached or go back to 1 if not.

This method can lead to layouts that do not reach the desired path length, whenever that happens the process just starts over until the desired path length works, similarly to a generate and test

method. Figure 3.13 illustrates the output of the process, the arrows indicate exits of the room, the "S" and "E" indicate Start and Exit respectively.

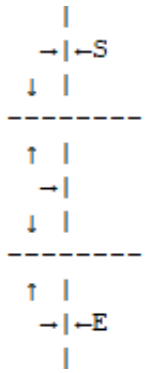
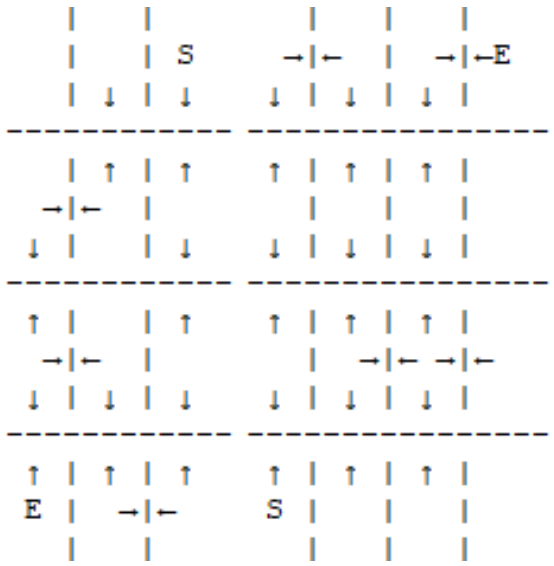


Figure 3.13: Example of the random walk process

After having the main path from the start to the exit of the floor, dead ends are considered. For that, the current layout is scanned in order to find possible dead ends. Dead ends are generated by going along the main path and looking for adjacent empty rooms, whenever such a situation arises, the algorithm marks that room as a possible dead end root. The same empty room can only have one adjacent dead end room to prevent dead ends from having multiple entrances since that would defeat the purpose. Start rooms and exit rooms do not count as possible dead end roots. Figure 3.13 features one possible dead end, with its root being the up/down arrow room.

After finding all the dead end roots, a count of the maximum possible dead ends is used to randomize the number of dead ends that end up being generated. If there is at least one possible dead end, then the range for the randomization is [1, maxDeadEnds], otherwise no dead ends are generated.

With the number of dead ends to generate decided, a random walk process similar to the main path generation is used to generate the dead ends. From each dead end root up to the number of desired dead ends, a random walk is performed until no more rooms can be created. To add variety, the dead ends do not always use the maximum amount of rooms they can, each room that composes a dead ends increases the chance that the dead end will stop early. Usually dead ends range from 1 to 3 rooms, due to the early stopping chance, and also dungeon area not having more than that number of contiguous empty rooms. Figure 3.14 shows examples of the output of this phase, sub-figure (a) shows an example of a dead end with more than 1 room of length, while sub-figure (b) shows an example of multiple 1 room length dead ends. Sub-figure (a) also shows that the dead end stopped early, since there is one more adjacent empty room available that was not used for the dead end. Sub-figure (b) also shows that the number of dead ends is randomized, the layout has 3 dead ends but could have a maximum of 4, one below the "E" room.



(a) (b)

Figure 3.14: Examples of the output after dead end generation

Having the layout of the dungeon decided, the remaining step is filling the empty rooms with content. For each empty room that is not a dead end, the algorithm runs a randomization of the content and the difficulty if it applies. Dead ends are always combat rooms and the only randomization is their difficulty. Table 3.7 shows the probability of each room, that is not a dead end, being a certain content type. For combat and puzzle rooms, the difficulty also has to be randomized, table 3.8 shows the probability of a certain difficulty being chosen given the current floor number. The probabilities for type of content and difficulty were balanced through personal testing of the game, and were meant to create a game experience of rising difficulty as the player progresses from the first floor to the fifth floor, peaking after the fifth floor.

Content Type	Probability
Combat	30%
Puzzle	30%
Corridor	40%

Table 3.7: Probabilities of choosing each content type

Floor Number	Easy	Normal	Hard
1	75%	25%	0%
2, 3	40%	50%	10%
4, 5	10%	60%	30%
>5	0%	25%	75%

Table 3.8: Probability of each difficulty per floor number

Figure 3.15 shows an example of the final generation output. The left side shows the layout and connections of the rooms, while the right side shows the content type, and difficulty of the rooms. Single "C" rooms are corridor rooms that only serve to connect rooms and break the monotony of constant combat/puzzle situations, "C," rooms are combat rooms and the second letter represents the difficulty (**E**asy, **N**ormal, **H**ard), finally, "P," rooms are puzzle rooms and the second letter is also the difficulty. The "S" and "E" indicate Start and Exit respectively. Appendix A shows an example of the full procedural generation log.

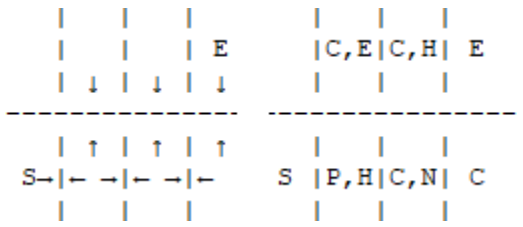


Figure 3.15: Example of a full floor generation

This concludes the detailing of the procedural generation process. The process is parameterized in a way that allows tuning values in order to modify the floor generation, the idea being that the adaptive version will tune these parameters in some way. For clarity, a summary of the procedural generation parameters follows:

- **Dungeon Size** - the number of possible rooms in the horizontal and vertical direction.
- **Shortest Path to Exit** - the shortest path length to reach the exit of the floor.
- **Dead Ends Total** - the total number of dead ends of the floor.
- **Combat Rooms** - the number of combat rooms and their location.
- **Puzzle Rooms** - the number of puzzle rooms and their location.
- **Corridor Rooms** - the number of corridor rooms and their location.

3.5.7 Logging User Data

For the implementation of the Machine Learning technique, as well as assessing its performance, logging of the users' performances was needed. As such the game creates a set of files whenever the player starts a new game, which are as follows:

- **YYYY-MM-DD hh mm ss ProcGen.txt** - contains the output of the procedural generation of each floor. Used for debugging reasons, not relevant to the objectives.
- **YYYY-MM-DD hh mm ss GUID.txt** - contains a Globally Unique Identifier (GUID) that identifies the play session. As the players have to email the log files to me and also fill an online form, this GUID is used to distinguish between players and to link the email to the form in an anonymous way.

- **YYYY-MM-DD hh mm ss Log.csv** - contains all the gameplay variables that are logged such as floor size, floor time, health lost per floor, combat performance, puzzle performance and others. Main files used for evaluation and creating the adaptive version of the game.

The comma-separated values (CSV) file, Log.csv, contains the user data that is relevant to the evaluation process and to the implementation of the adaptive version. Details about each column of the log can be found in section 4.2. For now, the implementation of the adaptive version follows in section 3.6 and explains how these log files were used to create it.

3.6 Implementing the Adaptive Game

For creating the adaptive version, 3 methods were considered: using k-NN, using Unity ML Agents, or using a Recommender System. The idea was that from a first batch of user tests which yielded results, the results could be used as a basis for adaptation. The adaptation would have to rely on altering the procedural generation parameters and to do so, it only had the user logs from the first batch of user tests and the corresponding filled forms. Unfortunately, this limited the available methods to use for adaptation as the following paragraphs will detail.

The recommender system method, described in subsection 2.1.4, works in a way that by knowing the current user's tastes, it can recommend items from users with similar tastes. For this to work, the current user has to be profiled and via that profiling their similarity to other users computed so that the system can recommend items. In the adaptive version of the game, this system would work by suggesting floor layouts that have worked well for previous users by knowing the similarity between the current user and the previous users. However, since no profiling was done on the user tests nor is the current player profiled, there is no way of computing similarity and as such this method can not be used. Using recommender systems only came about after the game was already developed, as such decisions made beforehand impacted the viability of using this method.

The Unity ML Agents method allows a lot of machine learning techniques to be applied, but the most important one for implementing the adaptive version would be reinforcement learning, described in subsection 2.1.1. The idea would be to task the agent with the generation of the floors and reward it on completion. For example, by rewarding floors where the player took less time than the average, lost less health points than the average and also rewarding content density so that the agent avoided placing only empty rooms. The average time for floor completion and puzzles, as well as the average health points lost per floor could be calculated from the first batch of user tests. However, Unity ML Agents requires the training phase to be run as a simulation which means no manual input can be performed making it not suitable for the adaptive version that was envisioned. The player character would have to be simulated and their inputs through the whole floor simulated, for that to work an agent would have to be trained to play the game while another agent generated the floors. Due to time constraints and to not having enough user data to reliably train an agent to play the game that would perform similar to real humans, this option

was discarded. This was the original method planned and the one that was considered the best contender for improving the game experience when compared to the non-adaptive game.

The k-NN method, described in subsection 2.1.3, works by taking input data and matching it to a set of labeled data. The k indicates how many closest matches the algorithm should return. Usually the algorithm is used to classify new data points by matching it to the closest k labeled data points, which are already assigned a class. The class of the new data point is, usually, the most common class from the k nearest matches. While classification is not the purpose of the adaptive version, the k-NN method can be used to match the current player to previous players from the first batch of user tests. This method was the implemented one, due to time constraints not allowing the RL method and lack of profiling not allowing the recommender system method.

For the implementation of the k-NN method, the Game Experience Questionnaire (GEQ) results from the first batch of users tests were used. The user logs were pruned so that only logs that corresponded to good GEQ results were left. The GEQ returns several measures and for the pruning, scores greater than 2.00 on negative measures were pruned, and scores lower than 2.00 on positive measures were pruned. The rationale being that these values are the middle point in the GEQ, which ranges from 0 to 4 on its measures. Combining these logs, the labeled data to which the k-NN algorithm could match to was created. By playing the adaptive version of the game, the current player's floor performance was matched to the previous users' floor performances for the 3 closest matches, using Euclidean distance. The generation of the next floor is decided from the closest match, giving a floor with similar parameters to what the user had played. Since the users from the first batch of user tests were pruned so that only the ones that had good experiences with the game remained, the idea was that by matching floor performance, the new player could have a similar experience. Results analysis will determine if this did indeed work as intended.

For choosing which fields to use for the matching process, the fields that had more impact on the generation parameters were picked, along with all the fields that indicated player performance. For clarity, the process used the following user log fields:

- Dungeon Area
- Shortest Path to Exit
- Times Minimum Path Traversed
- Floor HP Lost
- Percent Time Spent in Combat
- Percent Time Spent in Puzzles

To break ties, in cases where more than one match were at the same distance, the following user log fields were used:

- Dead Ends Explored %

- Times Minimum Path Traversed

The tiebreaking worked by finding the lowest values of the presented fields since it would lead to floors with less backtracking. The matching process converted the values from whatever numeric range they used to a range of [0, 1] so that no field had more priority than others when using the k-NN method. Figure 3.16 shows the integration of the k-NN method with the game's components.

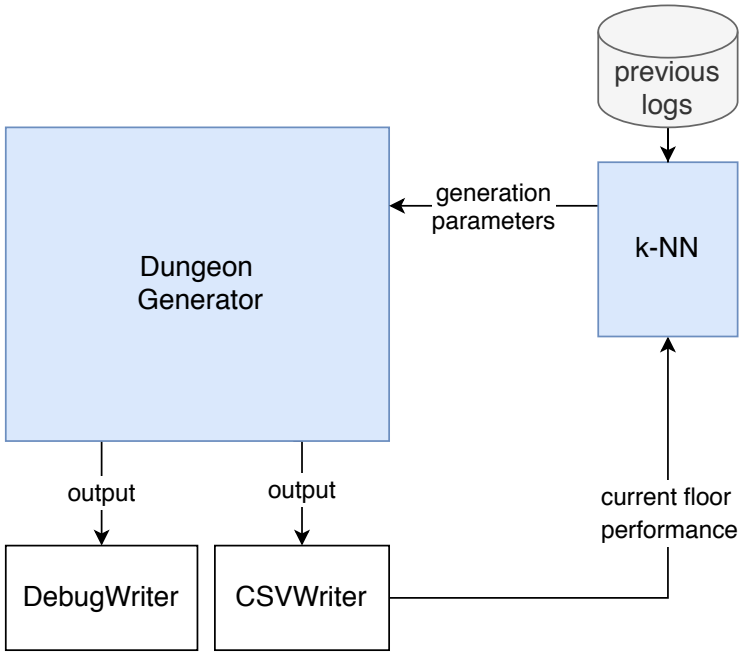


Figure 3.16: Integration of the k-NN method with the game's components

Having implemented an adaptive version of the game, a second batch of user tests was performed and the analysis and discussion of the results, along with the main conclusions on whether the method fulfilled the objectives can be found in chapter 4.

3.7 Summary

In this chapter, the concept and implementation of the proposed solution were presented. Section 3.1 detailed in a conceptual way what a generic adaptive game system would look like. The main idea being that a game should be parameterized in a way that allows fine tuning of its various components from an external source, allowing the usage of an adaptation algorithm that modifies the game in order to fulfill whatever objective is set (usually improve game experience for a particular player). Section 3.2 narrowed down the implementation specifics so as to reduce development time, since developing a generic solution would lead to a very complex framework. The game genre was set as 2D Top-Down Action and the adaptation focused on tuning of the procedural generation parameters.

Sections 3.3 and 3.4 described the architecture and tools used to implement the proposed solution. Section 3.5 detailed the game itself with focus on its gameplay, the user interface, the combat and puzzle segments, the procedural generation process and its parameters, as well as the data logged in order to aid the adaptation process. Finally, section 3.6 related the adaptation process and how its decisions were made. The final implementation uses k-NN to match current user performance to previous users' performances in order to provide a floor that is similar to the ones similar users had previously liked.

Chapter 4

Evaluation

User experiments were planned to analyze the performance of the implemented solution. An evaluation method was proposed and enacted. The idea was to evaluate whether the adaptive game developed using Machine Learning techniques had an impact on game experience. For that, a first batch of users tested the game without adaptive elements, and at a later date, a second batch of users tested the adaptive version of the game. Since users had to email the files, it its known that 2 users were present in both batches but every other user was unique.

This chapter starts by detailing the test protocol and structure in section 4.1, followed by a description of the types of data gathered throughout the user tests in section 4.2. Section 4.3 presents the obtained results and section 4.4 analyzes and discusses the presented results.

4.1 Test Protocol and Structure

Evaluation was divided into two batches, with the only difference being that one batch tested the non-adaptive game and another batch tested the adaptive game. Both batches did not know which version they were playing. Since the test was done remotely, potential testers received an email with instructions on how they could participate. The testing procedure proceeded along the following items:

1. The user decided to participate and the email directed them to the game's page[1] on itch.io so that they could read further instructions and download the game for their OS.
2. After playing the game (adaptive or not), users were instructed to email the "UserLogs" folder to me, these contained the data gathered throughout gameplay.
3. Finally, users were instructed to fill a short form about their play session. Users were given a unique identifier to use in the form so that the email they sent could be linked to the form they filled in without revealing personal details.

[1] https://ezspecial.itch.io/dungeon-escape

Test data was gathered during the users' play sessions and also via the form that they filled in at the end of the session. This data is meant to be used to compare the two batches of users and check the impact of the adaptive version when compared to the non-adaptive version. Section 4.2 details the gathered test data.

4.2 Gathered Data

As previously mentioned, to assess the performance of the chosen Machine Learning technique, data was gathered so that the two batches of user tests could be compared. This data is comprised of logs that the game automatically generates, and of a form that users filled in at the end of the test session. In order to make the results section clearer, this section describes the gathered data.

Starting with the logs, they are timestamped with date and time (YYYY-MM-DD hh mm ss format) and each set of logs belongs to a play session. A play session is the sequence of events from the start of the game through the main menu, up to the game over screen or the player quitting the game. For the same player, each new game creates a new play session. Players could send any of the play sessions they desired since a GUID is created per play session and as such would count as a new entry when filling the form. The game generates 3 files: ProcGen.txt, GUID.txt and Log.csv. From these, Log.csv is the only relevant file for results. A brief description of the files can be found in section 3.5.7. This section elaborates upon the description of the Log.csv file since the other files are already fully described. The Log.csv file is divided into rows where the first row is the header, which describes the column contents, and the other rows store values per dungeon floor that the player reached. The final row serves as a total row. Appendix B shows an example of this log, while the following list describes each column of it:

- **Dungeon Floor** - the dungeon floor that the row pertains to.
- **Dungeon Size X / Dungeon Size Y** - number of possible rooms in the horizontal and vertical axis.
- **Dungeon Area** - the total area of the floor.
- **Shortest Path to Exit** - the shortest path to the exit, meaning the minimum number of rooms the player has to cross to get to the floor exit.
- **Dead Ends Explored / Dead Ends Total** - the number of dead ends the player reached, along with the total dead ends of the floor. A percentage of dead ends explored is also logged.
- **Rooms Explored / Room Total** - the number of rooms explored by the player, along with the total room number of the floor. A percentage of rooms explored is also logged.
- **Rooms Traversed** - the number of rooms the player traversed on the floor, counting repeat rooms. This number is meant to serve as an indicator on whether the player was lost.

- **Times Minimum Path Traversed** - the number of times the player traversed the minimum possible path, given by the ratio $RoomsTraversed/ShortestPathtoExit$. This indicates how many more rooms the player explored than needed to finish the floor.
- **Floor HP Lost** - the amount of health points (HP) the player lost on the floor.
- **Would die old HP? / Died?** - boolean values that relate whether the player would have died on the floor with the old HP total (6 HP), or whether they died with the current HP total (10 HP).
- **Quit?** - self-explanatory, whether the player quit the game by closing the game window.
- **Time Spent in Combat** - total seconds spent in combat situations on the floor, also has a column with the percentage of time spent in combat.
- **Time Spent in Puzzles** - total seconds spent in puzzles on the floor, also has a column with the percentage of time spent in puzzles.
- **Floor Time Taken** - total seconds taken for the whole floor. Used to calculate percentages for the other columns.
- **# Easy Combat / # Normal Combat / # Hard Combat** - the number of combats that belonged to a particular difficulty category on the floor.
- **Total Combat # / Combat Room %** - the total number of combat rooms on the floor as well as the percentage of combat rooms.
- **# Easy Puzzle / # Normal Puzzle / # Hard Puzzle** - the number of puzzles that belonged to a particular difficulty category on the floor.
- **Total Puzzle # / Puzzle Room %** - the total number of puzzle rooms on the floor as well as the percentage of puzzle rooms.
- **# Corridors / Corridor Room %** - the total number of corridor rooms on the floor as well as the percentage of corridor rooms.
- **Total Content Rooms** - the total number of rooms on the floor, excluding the start and exit rooms. Used to calculate the percentage of combat, puzzle and corridor rooms per floor.
- **Room Content NxN** - the content of the room with coordinates (N, N).
- **Room Difficulty NxN** - the difficulty of the room with coordinates (N, N).
- **Room HP Lost NxN** - the HP lost in the room with coordinates (N, N).
- **Room Solution Time NxN** - time taken in the room with coordinates (N, N).
- **Room Explored NxN** - whether the room with coordinates (N, N) was explored by the player.

The last section of columns of the Log.csv file log certain values in respect to each individual room instance, the columns always exist and go from 0x0 to 3x3 (a max of 16 rooms). Whenever a room does not exist in the floor itself, the columns are simply blank for that particular floor.

Besides the automatically generated logs, users also filled a form. This form contained a small amount of questions related to age group, frequency of gaming and whether the player knew about the genre they had just played. The form then administers the Game Experience Questionnaire [60]. From the GEQ, the Core and Postgame modules are used, which are meant to measure the experience during the play session and after the play session is over. The form can be seen in appendix C.

With the description of the gathered data finished, the results can now be presented and then discussed.

4.3 Results

The results were gathered in two batches, both batches being composed of students and ex-students of the University of Porto[2]. From those, most are currently attending or have attended its faculty of engineering[3]. Table 4.1 shows how many unique users participated as well as the distribution of users that only submitted the user logs, only filled the form, and the ones that did both.

Batch	Only Sent Logs	Only Filled Form	Did Both	Total Unique Users
Default	4	7	30	41
Adaptive	6	2	39	47

Table 4.1: Number of users and what they submitted per test batch

This section will show the results obtained from both batches, starting with the demographics and gaming habits questions on the form in subsection 4.3.1. Results from the user logs follow in subsection 4.3.2, ending with the GEQ results in subsection 4.3.3. A small analysis will be performed per subsection with a overall analysis and discussion in section 4.4. It is important to notice that the first batch tested the non-adaptive game, while the second batch tested the adaptive game.

4.3.1 Form Questions

The form had a small set of questions about age and gaming habits which were meant to serve as a way of knowing whether these factors might have had an impact on the results. The questions were as follows:

- What is your age group?

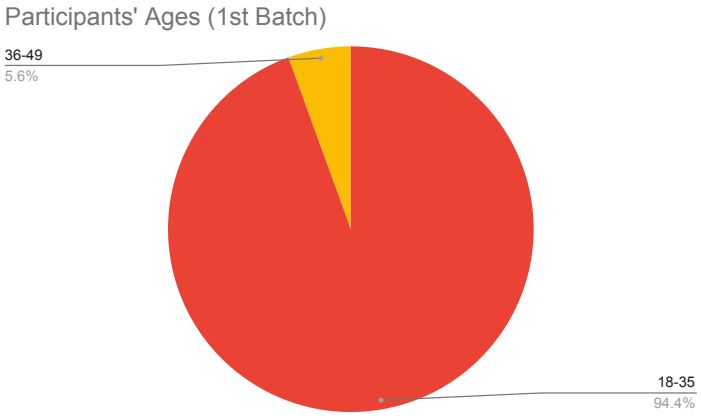
[2] https://sigarra.up.pt/up/en/WEB_BASE.GERA_PAGINA?p_pagina=home
[3] https://sigarra.up.pt/feup/en/web_page.inicial

- Do you play videogames regularly? (at least 6h per week)[4]
- Have you played games similar to this one?

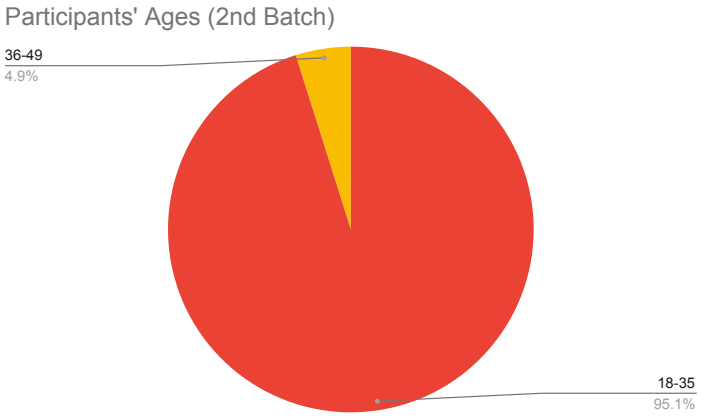
Figure 4.1 shows the distribution of the participants' ages in both batches. As expected the most common age group was 18 to 35 years old, since most participants were currently attending university. As for gaming habits, the first batch had an almost even distribution of users that played games regularly and users that did not. However, the second batch had an increase in users that played games regularly, raising the split to 75/25 instead of 50/50, as Figure 4.2 shows. Besides playing games regularly, most (>80%) users from both batches played this genre of game as can be seen in Figure 4.3.

Those are the results regarding the three form questions, additionally a stacked graph was made to understand how many users played this type of game inside the group of users that play games regularly and the group of users who do not. In the first batch, both the group of users that played regularly and the group that did not, had similar splits regarding having played similar games before. Meanwhile, the second batch had similar results except that the group of users that did not play regularly had a higher ratio of users that had played similar games before. The stacked graph and what has been described can be seen in Figure 4.4.

[4] value taken from [61]

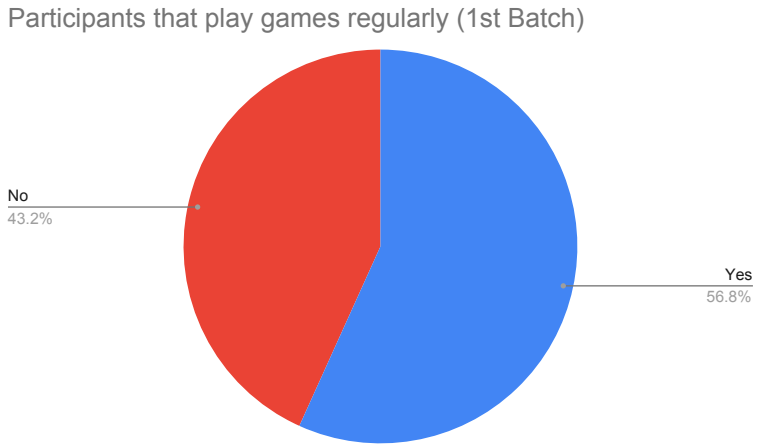


(a)

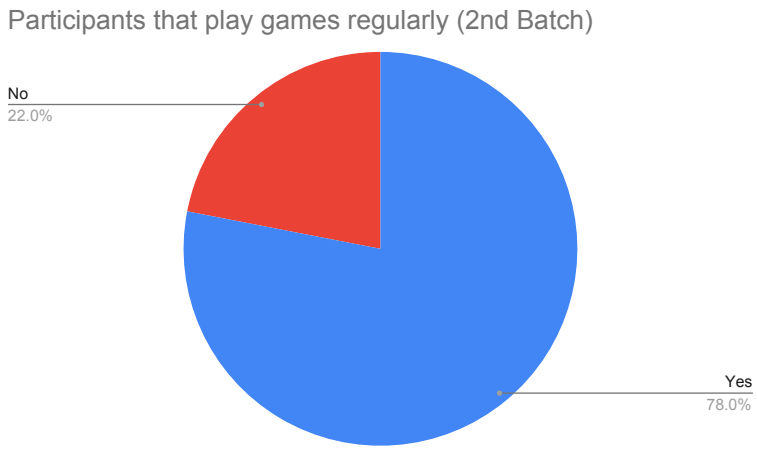


(b)

Figure 4.1: Age distribution of the first batch (a) and the second batch (b)

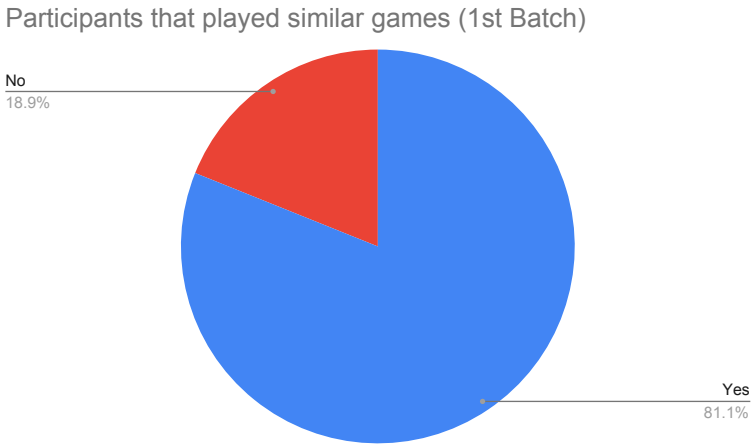


(a)

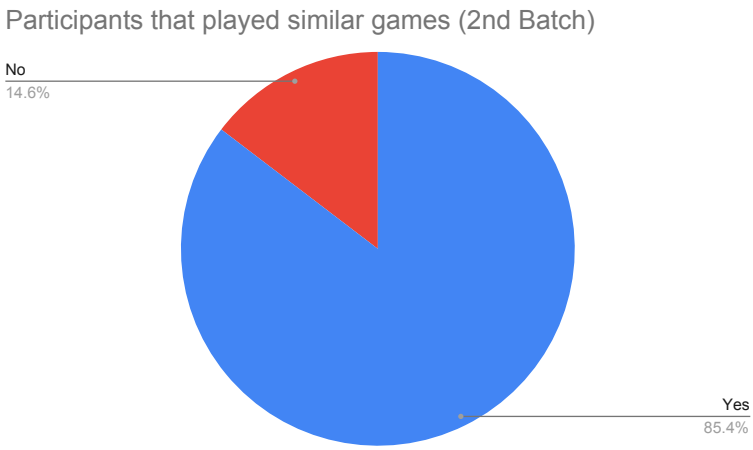


(b)

Figure 4.2: Percentage of users that play games regularly in the first batch (a) and the second batch (b)

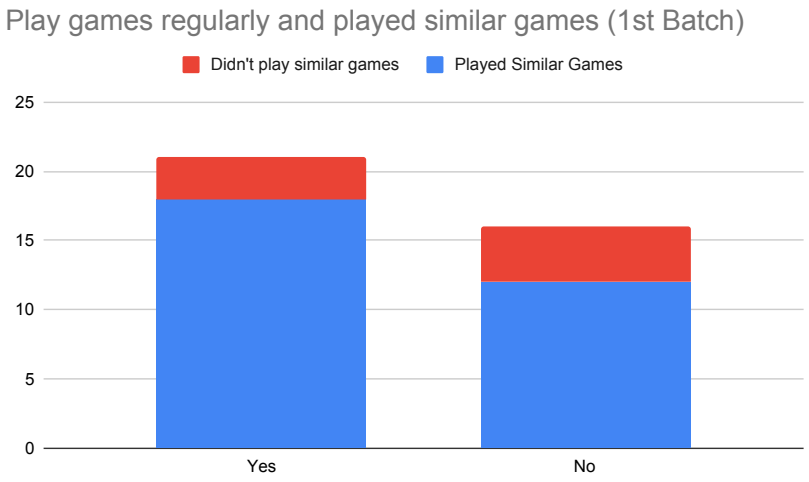


(a)

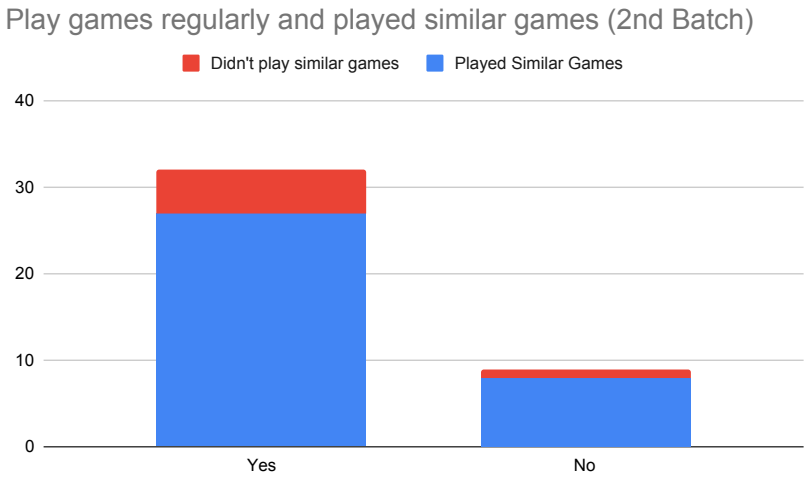


(b)

Figure 4.3: Percentage of users that had played similar games in the first batch (a) and the second batch (b)



(a)



(b)

Figure 4.4: Distribution of users that had played similar games inside the group of users that play regularly and the group of users that do not

4.3.2 User Logs Data

Users emailed their logs which contained all the data mentioned in section 4.2. For each batch, the total row of the user play session log was copied into a sheet with all the other total rows. Afterwards, the maximum value, minimum value, average value and median value of each column of the combined total rows was computed and is shown on table 4.2 for both batches. The following list details them one by one:

- **Max Floor** - the max floor each user got to. The maximum raised considerably on the second batch, as did the average and median.

- **Average Shortest Path** - the average shortest path to the exit for each user. The second batch had slightly shorter floors, with a lower maximum, average and median.
- **Average Dead Ends per Floor** - the average dead ends per floor for each user. No significant change between batches.
- **Dead Ends Explored (%)** - the percentage of dead ends each user explored. No significant (less than 10%) change between batches, most users end up exploring half of the total dead ends.
- **Average Rooms per Floor** - the average number of rooms per floor for each user. The second batch had less rooms per floor with a lower maximum, average and median. This is expected since the shortest path decreased on the second batch, which leads to less rooms being generated.
- **Average Rooms Traversed per Floor** - the average rooms traversed per floor for each user, also counting previously explored rooms. The second batch had a lower maximum, average and median. Also expected since floors are shorter on the second batch.
- **Average of Times Minimum Path Traversed** - the average number of times each user had to traverse the minimum path to the exit per floor. The second batch had an increase of the maximum, average and median values. This measure indicates how lost the users were.
- **Average HP Lost per Floor** - the average health points lost per floor for each user. The second batch had lower average and median. This could mean the game was easier on the second batch or that it had less combat rooms.
- **Time Spent in Combat (%)** - the percentage of time each user spent in combat situations. While the maximum of the second batch had a big increase, the average and median remain mostly unchanged.
- **Time Spent in Puzzles (%)** - the percentage of time each user spent on puzzles. The second batch had a lower average and median, almost half of the first batch.
- **Average Floor Time (s)** - the average time to clear a floor for each user. The second batch had a maximum and minimum at almost half of the first batch. Besides that, the average and median were considerably (25%) lower on the second batch.
- **Combat Room (%)** - the percentage of combat rooms each user could run into. The second batch had an increase (around 5 points on the percentage) of the average and median.
- **Puzzle Room (%)** - the percentage of puzzle rooms each user could run into. The second batch had a lower average and median. The additional combat rooms on the second batch most likely caused the drop in puzzle room number, since corridor rooms remain mostly the same percentage.

- **Corridor Room (%)** - the percentage of corridor rooms each user could run into. Almost unchanged between batches.

Besides these measures, the method through which each user ended the play session was also logged. Table 4.3 shows how many users per batch died or quit the game. The first batch had a split of 80/20, 80% being the users that died and 20% being the users that quit. The second batch was more even with a split of 65/35, meaning a lot more players decided to quit instead of continuing until a game over.

	First Batch				**Second Batch**			
Statistic	**Max**	**Min**	**Avg**	**Median**	**Max**	**Min**	**Avg**	**Median**
Max Floor	14	1	5.65	5.50	65	1	12.71	8.00
Average Shortest Path	9.29	4.00	6.78	6.83	8.62	3.50	5.22	5.20
Average Dead Ends per Floor	2.00	0.50	1.25	1.27	2.00	0.00	1.01	1.00
Dead Ends Explored (%)	100.00	0.00	50.26	53.50	100.00	0.00	52.47	50.00
Average Rooms per Floor	11.07	5.33	8.23	8.29	10.10	4.50	6.36	6.20
Average Rooms Traversed per Floor	13.67	2.00	7.89	8.13	9.95	3.75	6.52	6.40
Average of Times Minimum Path Traversed	1.82	0.50	1.16	1.08	1.94	0.80	1.25	1.21
Average HP Lost per Floor	10.00	2.00	4.67	4.61	10.00	0.92	3.77	3.35
Time Spent in Combat (%)	48.00	0.00	32.21	34.50	60.00	9.00	35.00	35.00
Time Spent in Puzzles (%)	53.00	0.00	19.79	20.00	41.00	0.00	13.20	13.00
Average Floor Time (s)	215.49	42.73	81.72	75.53	129.80	24.10	58.03	51.11
Combat Room (%)	75.00	29.00	46.41	45.00	90.00	30.00	52.00	50.00
Puzzle Room (%)	38.00	0.00	22.68	23.00	40.00	0.00	17.33	20.00
Corridor Room (%)	46.00	13.00	30.85	32.00	50.00	10.00	30.67	30.00

Table 4.2: User log statistics from both batches

Statistic	**First Batch**	**Second Batch**
Died?	27 (79%)	30 (66%)
Quit?	7 (21%)	15 (34%)

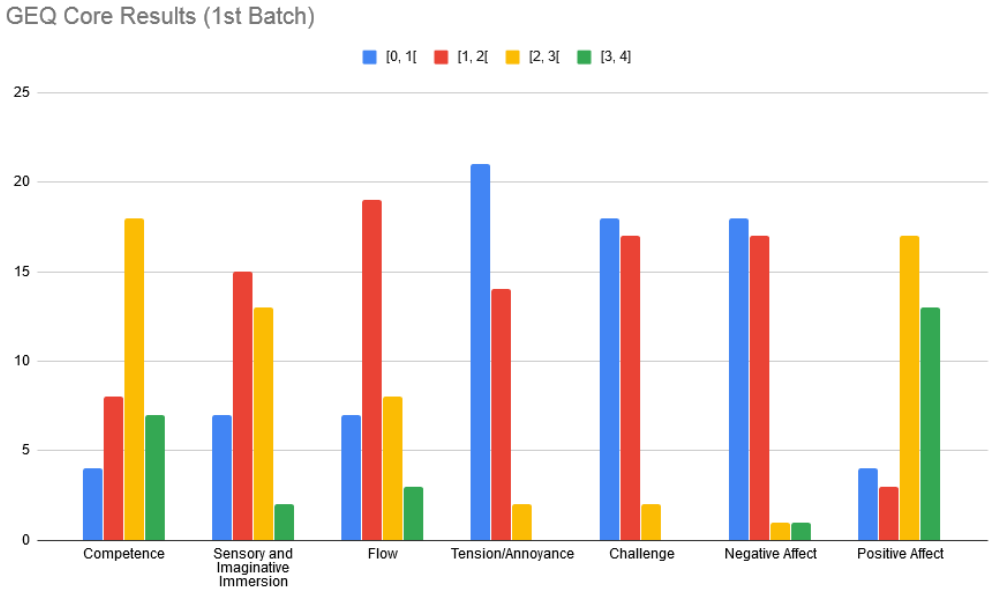
Table 4.3: How did users end their play session for both batches

4.3.3 Game Experience Questionnaire

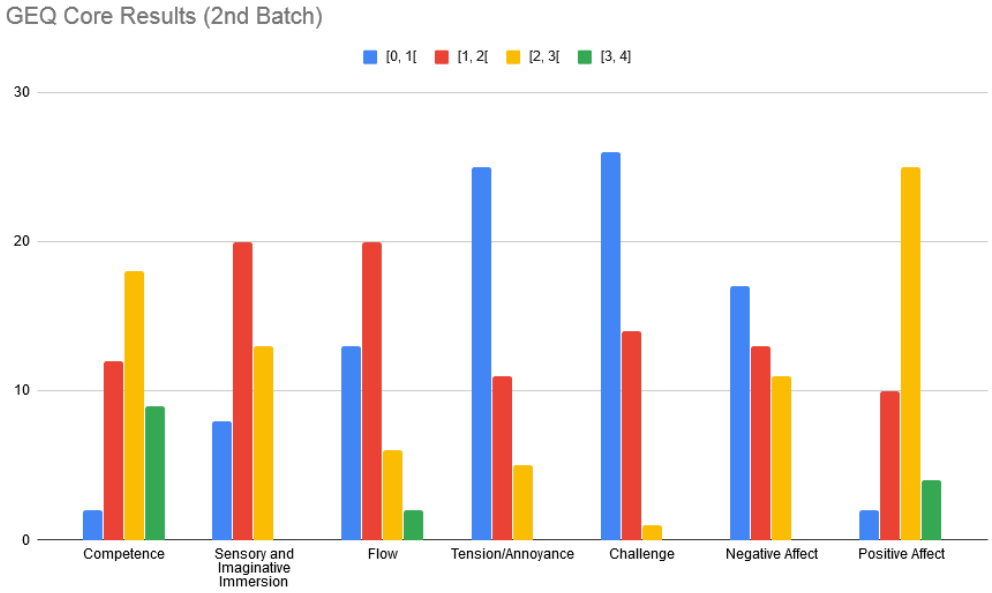
The GEQ Core and Postgame modules were administered on the form the users filled in. The GEQ Core module measures game experience during gameplay, while the GEQ Postgame module

measures game experience after the play session is over. Both modules have questions that have scores from 0 to 4 (not at all, slightly, moderately, fairly, extremely) and belong to a certain category. At the end, each category is tallied and averaged to give a standard score between 0 and 4. The core module is divided into 6 categories: competence, sensory and imaginative immersion, flow, tension/annoyance, challenge, negative affect, and positive affect. The postgame module is divided into 4 categories: positive experience, negative experience, tiredness, and returning to reality.

Regarding the results, two bar graphs showing the distribution of results for each category for each batch can be seen in Figure 4.5 for the core module, and Figure 4.6 for the postgame module. For a more accurate representation of the data, a table showing the average and median of each category for each batch can be seen on table 4.4 for the core module, and table 4.6 for the postgame module. For analysis, a table showing the change of the average and median from the first to the second batch for each category was made for each module. Table 4.5 shows the change for the core module, and table 4.7 shows the change for the postgame module. At a glance, there is little difference between the non-adaptive and adaptive versions.



(a)



(b)

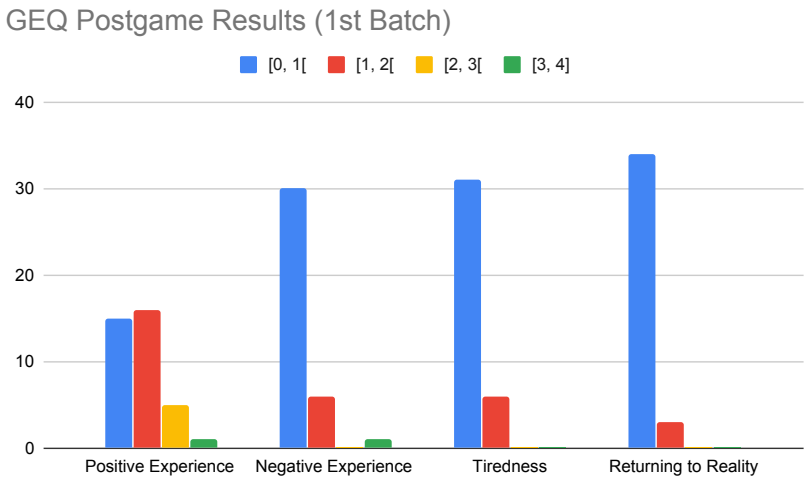
Figure 4.5: GEQ Core results for the first batch (a) and second batch (b)

	First Batch		Second Batch	
Statistic	**Average**	**Median**	**Average**	**Median**
Competence	2.11	2.40	2.27	2.40
Sensory and Imaginative Immersion	1.76	1.83	1.50	1.50
Flow	1.56	1.40	1.33	1.20
Tension/Annoyance	0.67	0.67	0.67	0.33
Challenge	0.99	1.00	0.71	0.60
Negative Affect	0.99	1.00	1.26	1.00
Positive Affect	2.36	2.40	2.16	2.20

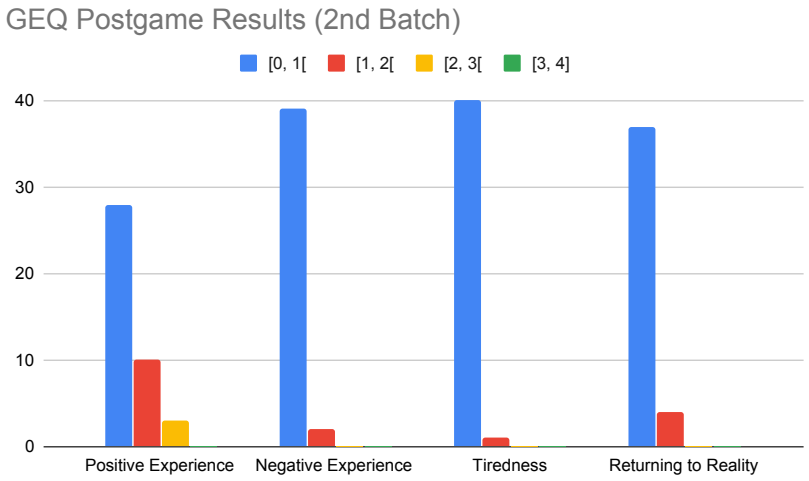
Table 4.4: GEQ Core average and median values for each component of both batches of users

	Change From First to Second Batch	
Statistic	**Average Change**	**Median Change**
Competence	0.16	0.00
Sensory and Imaginative Immersion	-0.26	-0.33
Flow	-0.23	-0.20
Tension/Annoyance	0.00	-0.34
Challenge	-0.28	-0.40
Negative Affect	0.27	0.00
Positive Affect	-0.20	-0.20

Table 4.5: GEQ Core change from first to second batch of users



(a)



(b)

Figure 4.6: GEQ Postgame results for the first batch (a) and second batch (b)

	First Batch		Second Batch	
Statistic	**Average**	**Median**	**Average**	**Median**
Positive Experience	1.18	1.17	0.78	0.83
Negative Experience	0.50	0.33	0.36	0.33
Tiredness	0.27	0.00	0.11	0.00
Returning to Reality	0.31	0.33	0.23	0.00

Table 4.6: GEQ Postgame average and median values for each component of both batches of users

	Change From First to Second Batch	
Statistic	**Average Change**	**Median Change**
Positive Experience	-0.40	-0.34
Negative Experience	-0.14	0.00
Tiredness	-0.16	0.00
Returning to Reality	-0.08	-0.33

Table 4.7: GEQ Postgame change from first to second batch of users

4.4 Analysis and Discussion

This section will focus on analyzing the results and discussing what they could mean in the context of the overall objective, that being the improvement of the game experience using a ML technique. Subsection 4.4.1 focuses on the form questions about age group and gaming habits. Subsection 4.4.2 focuses on the user logs that were gathered and what can be concluded from them. Finally, subsection 4.4.3 focuses on the Game Experience Questionnaire and its results, which allow concluding whether the objective was achieved.

4.4.1 Form Questions

From the questions pertaining to demographics and gaming habits, it can be seen that the age groups were mostly the same on both user batches. The users that played games regularly increased from around 50% to around 75% on the second batch, while the users that had played similar games before were about the same on both batches. The stacked graph shown in Figure 4.4 shows that whether the users played games regularly or not, they still had played a similar game before. No particular difference can be seen between the users that played games regularly and the ones that do not. From an evaluation standpoint, not much can be retrieved from just this information. Being a group of users that were knowledgeable of games and, particularly, these types of games might lead to the users being more critic of the game, since they have something to compare it to. Further tests would have to be made regarding users that do not play games regularly, since that group is underrepresented in the user tests. Another underrepresented group is age groups outside the 18 to 35 years old age group. This might mean that generalizing the results for users that do not play regularly, or other age groups, might not be valid. In terms of age groups, the game experience wildly varies with lived experiences, so older age groups might be less impressionable and have a very different response to the stimuli the game provides.

4.4.2 User Logs Data

Regarding the user logs data analysis, the maximum floor reached on the adaptive version was a lot higher (4.6 times higher) than the non-adaptive version, with its average being more than the double of the non-adaptive version. This might mean that the game was made easier by the

adaptive process, or that floors were shorter and quicker to complete. The average shortest path measure confirms this assumption about the floors, since they had about 1 room less on average on the adaptive version. Average dead ends per floor and percentage of dead ends explored saw no significant changes, both batches of users usually ending up exploring half of the possible dead ends. Average rooms per floor and average rooms traversed per floor decreased in the adaptive version. Users had around 2 less rooms to explore per floor on average and ended up traversing about 2 less on average too. This makes sense since each floor on average had lost 2 rooms. The average number of times users traversed the minimum path slightly (7%) increased in the adaptive version, but the increase is small enough to be chalked up to coincidence. Users ended up visiting around 20% more rooms than they would need to finish the floors with minimum rooms traversed, which is due to the dead ends they end up visiting. Average health points lost per floor lowered in the adaptive version, dropping almost 1 point in both the average and the median. This could mean that the game was made easier in the adaptive version, or also that there were less combats or at least easier ones. The percentage of time spent on combat rooms was about the same on both versions, while the percentage of time spent on puzzles was about half (average and median) in the adaptive version. This could mean the puzzles were easier on the adaptive version, or that the adaptive version had less puzzle rooms. The average time, in seconds, to complete each floor was 25% lower in the adaptive version when considering the average and the median. This could be due do the shorter floors mentioned previously, but combats or puzzles might also have been easier in the adaptive version. The adaptive version had about 5% more combat rooms than the non-adaptive version, with the same percentage dropping on the amount of puzzle rooms in the adaptive version. Corridor rooms remained the same on both versions.

From the analysis of the user logs, it can be argued that the adaptive version was made easier. The average maximum floor reached increased in the adaptive version, and the average health points lost dropped by 1 point in the adaptive version, which points to an easier experience. The percentage of time spent in combat rooms remained the same but the average time to complete floors was 25% less, on average, in the adaptive version, which means that the adaptive version users spent less absolute time on combat situations. The combat room percentage remained the same but floors were shorter in the adaptive version, meaning that the absolute number of combat rooms was also lower, possibly leading to the faster completion of floors and less health points lost. In summary, it seems that the adaptive version made the game easier and more streamlined; whether this resulted in a better experience will be measured by the GEQ.

4.4.3 Game Experience Questionnaire

The GEQ results are the last analysis that can be performed. The GEQ will directly answer whether the intended objective of the work was achieved. To remember that GEQ scores range from 0 to 4. Looking at the core module first, the competence category was about the same on both batches (>2). Being higher than 2 means that users felt moderately competent when playing both versions. For the sensory and imaginative immersion category, users were also similarly stimulated by the game on both batches (1.65). A score of 1.65 means that users were moderately

stimulated, but not as intense as they felt competent when playing. Users felt 15% less flow when playing the adaptive version, meaning users that played the non-adaptive version were more "in the zone" so to speak. The flow component score (1.50) corresponds to users slightly feeling flow. This decrease in flow might be due to floors being shorter and as such, the period between floors where players are forced to unwind was more frequent in the adaptive version. Users felt similar tension and annoyance on both batches (0.67), meaning they mostly did not feel tense nor annoyed. On the challenge category, the adaptive version had about 30% less score on this component which indicates that the assumption that the game was made easier in the adaptive version is probably correct. The score of the challenge category was around 1, which means users felt slightly challenged. Regarding negative affect, users felt 26% more negative affect in the adaptive version. With a score of around 1, negative affect was only slightly felt by the users. Finally, on the positive affect component, users felt 9% less positive affect in the adaptive version. Its score of around 2, means that users felt a moderate amount of positive affect. As for the postgame module, users felt 34% less of the game as a positive experience but also 30% less of the game as a negative experience. Both versions were considered as slightly positive experiences (score of 1), and both were considered not to be negative experiences (score of almost 0). Tiredness decreased by 60% in the adaptive version but was already extremely low (0.27) in the non-adaptive version, meaning that the game was not tiring at all. Finally, both versions were easy to return to reality from (scores of almost 0), with the adaptive version being 26% easier on average.

The GEQ shows that the adaptive version made users feel more competent and less challenged, but in doing so the flow and positive affect also decreased. As for the feelings of the users postgame, they felt less of a positive experience in the adaptive version but also less of a negative experience in that same version. The values of the postgame module show that both versions failed to leave a strong impression on its players, while the values of the core module show that players, ultimately, enjoyed (positive affect) the game while playing it. Due to the results obtained from the GEQ, it can not be said that the adaptive version provided a better game experience. Although it did provide an easier one that made users feel more competent, which was not the objective of the work. This might show that the method chosen was not a good fit, or that it was not applied in a way that would make it impactful.

Chapter 5

Conclusions

Adaptive games try to make regular games more appealing to a given player by taking into account the type of player that is currently playing, their preferences, or their performance. Due to the sheer number of people considered gamers nowadays, it becomes important to cater to the highest number of segments possible in order to better market games. This adaptation can be done to different types of content: the game world, the game mechanics, the AI, the narrative, or the scenario. Besides the different types of content that can be adapted, the objective of the adaptation is also very customizable. The goal might be to make the game more accessible difficulty wise, so that less skilled players can also enjoy the game. Others might want the game to be more engaging, adapting its narrative to the story preferences of the user. The possibilities are nearly endless when it comes to all the possible combinations of type of content and goal. As such, this thesis proposed the creation of an adaptive game using a machine learning technique for that adaptation process. The goal being whether the game experience could be improved given a specific technique.

With the objective in mind, chapter 2 focused on researching the state of art of machine learning techniques and also the game adaptivity field, which was the main contributing field. Several techniques were researched, including k-NN, reinforcement learning, recommender systems, and imitation learning. As for the game adaptivity, the adaptable content and player modeling was researched. A final section listed examples of game adaptivity using machine learning techniques. Knowing the techniques available and what adaptivity consisted of were crucial in proposing a solution.

After the research, a solution was proposed, which consisted of building a non-adaptive game and then adapting it with a machine learning technique. The developed game was a 2D Top-Down Action game that takes place in a dungeon with infinite floors. The floors are filled with puzzles and combats, the goal being reaching the exit on each floor until a game over or the player quits. For the adaptation, the procedural generation of the floors was parameterized and tuned by a k-NN algorithm, which tried to match the current player performance to previous players' performances.

The idea was to provide a similar floor to the one that the matched player had, hoping that the game experience would approach the one experienced by the previous players.

With the adaptive game implemented, there was a need to evaluate it in order to determine whether the objective of the thesis was achieved. For this, two user tests were done, one for the non-adaptive game and one for the adaptive game. Users had to fill a form which asked about their age and gaming habits, and then administered the Game Experience Questionnaire. Users also emailed log files from their play sessions. Results were that most people belonged to the same age group (18-35), played games regularly, and were familiar with the type of game developed. As for the user logs, they showed that the adaptive game seemed to make the game easier, making users reach a deeper floor than the users from the non-adaptive game. Finally, the GEQ showed that the adaptive version felt less challenging to the users, while making them feel more competent. However, it made no significant impact in positive or negative experience.

To conclude, the objective was not reached as the adaptive game did not improve upon the game experience of its users, since it had no significant increase on the positive affect or positive experience felt. However, it did succeed in making the game easier and users felt more competent when playing. Had this been the objective, it could be called a success. Without further user tests it would be hard to ascertain why the k-NN method did not achieve a better game experience, as it is hard to assess whether the fault was in the method or in the game itself not being parameterizable enough to really make a change. The choice of content to adapt might have also been the wrong one for the objective. ML-powered adaptivity seems to not have an universal solution that can lead to an improved experience. The next section suggests some future work topics that could lead to reaching the objective of the thesis.

5.1 Future Work

There are several approaches that could be done in order to further develop this work. For example, the test group was essentially composed of people aged 18 to 35 years old. Besides that, they were also mostly people that played games regularly and had played a similar game to the one developed. A possible work item would be to retry the test with a more heterogeneous group to check if the conclusion is the same for such a group. As it is, it can not be generalized that the results would be the same for other age groups or other gaming habits, since those have an impact on game experience.

The problem with the adaptive version might be on its foundation, the non-adaptive game, people seemed to like the combat segments better than the puzzles ones. Puzzle variety is indeed very low (6 total unique puzzles) in the game, which means that there is a possibility of adding a procedural generation component to the puzzles so that they are generated on the fly. This would let the adaptive game also control the puzzle generation which could lead to a better game experience. Obviously, it would mean that both versions would have to be tested again, since the non-adaptive version would also be changed.

A more drastic change to the implementation would be to implement a Reinforcement Learning adaptation method using Unity ML Agents for instance, which was not done due to time constraints. In that scenario, it would be possible to train an agent to play the game and then use that agent to train another agent that generates the levels. The idea being that the agent generating the levels would be trying to maximize player engagement. This maximization would be a topic of research since it is not clear cut how it could be done.

References

[1] R. Lopes and R. Bidarra, "Adaptivity challenges in games and simulations: A survey," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 3, no. 2, pp. 85–99, 2011.

[2] Remedy Entertainment, "Max payne." https://www.rockstargames.com/maxpayne/splash.html, 2001. Accessed: 2020-02-09.

[3] J. Walker, "Left 4 dead 2: Exclusive rps hands-on preview." https://www.rockpapershotgun.com/2009/06/01/left-4-dead-2-exclusive-rps-preview/, 2009. Accessed: 2020-02-09.

[4] Maxis, "Spore." https://www.spore.com/, 2008. Accessed: 2020-02-09.

[5] K. Compton and M. Mateas, "Procedural level design for platform games," *Proceedings of the 2nd Artificial Intelligence and Interactive Digital Entertainment Conference, AIIDE 2006*, pp. 109–111, 2006.

[6] J. Westra, F. Dignum, and V. Dignum, "Keeping the trainee on track," *Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games, CIG2010*, pp. 450–457, 2010.

[7] S. Bakkes, P. Spronck, and J. Van Den Herik, "Rapid and reliable adaptation of video game ai," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 1, no. 2, pp. 93–104, 2009.

[8] Entertainment Software Association, "2019 essential facts about the computer and video game industry." https://www.theesa.com/esa-research/2019-essential-facts-about-the-computer-and-video-game-industry/, 2019. Accessed: 2020-02-09.

[9] D. Michie, ""Memo" Functions and Machine Learning," *Nature*, vol. 218, no. 5136, pp. 19–22, 1968.

[10] R. S. Sutton, "Reinforcement Learning Architectures," in *Proceedings ISKIT'92 International Symposium on Neural Information Processing*, 1992.

[11] R. S. Sutton and A. G. Barto, "A temporal-difference model of classical conditioning," 1987.

[12] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, pp. 279–292, may 1992.

[13] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," *CoRR*, pp. 1–9, 2013.

[14] S. Thrun and L. Pratt, "Learning to Learn: Introduction and Overview," *Learning to Learn*, pp. 3–17, 1998.

[15] A. Lazaric, "Transfer in reinforcement learning: A framework and a survey," *Adaptation, Learning, and Optimization*, vol. 12, pp. 143–173, 2012.

[16] P. Taylor, Matthew E. and Stone, "Transfer Learning for Reinforcement Learning Domains: A Survey," *The Journal of Machine Learning Research*, vol. 10, pp. 1633–1685, 2009.

[17] M. G. Madden and T. Howley, "Transfer of experience between reinforcement learning environments with progressive difficulty," *Artificial Intelligence Review*, vol. 21, no. 3-4, pp. 375–398, 2004.

[18] N. Mehta, S. Natarajan, P. Tadepalli, and A. Fern, "Transfer in variable-reward hierarchical reinforcement learning," *Machine Learning*, vol. 73, no. 3, pp. 289–312, 2008.

[19] J. Ramon, K. Driessens, and T. Croonenborghs, "Transfer learning in reinforcement learning problems through partial policy recycling," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 4701 LNAI, pp. 699–707, 2007.

[20] G. Kuhlmann and P. Stone, "Graph-based domain mapping for transfer learning in general games," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 4701 LNAI, pp. 188–200, 2007.

[21] A. N. Meltzoff and M. K. Moore, "Imitation in newborn infants: Exploring the range of gestures imitated and the underlying mechanisms.," *Developmental Psychology*, vol. 25, no. 6, pp. 954–962, 1989.

[22] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation learning: A survey of learning methods," *ACM Computing Surveys*, vol. 50, no. 2, 2017.

[23] A. L. Friesen and R. P. Rao, "Imitation learning with hierarchical actions," *2010 IEEE 9th International Conference on Development and Learning, ICDL-2010 - Conference Program*, pp. 263–268, 2010.

[24] B. Piot, M. Geist, and O. Pietquin, "Bridging the gap between imitation learning and inverse reinforcement learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 8, pp. 1814–1826, 2017.

[25] A. P. Shon, D. Verma, and R. P. Rao, "Active imitation learning," *Proceedings of the National Conference on Artificial Intelligence*, vol. 1, pp. 756–762, 2007.

[26] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, pp. 21–27, January 1967.

[27] Z. Pan, Y. Wang, and Y. Pan, "A new locally adaptive k-nearest neighbor algorithm based on discrimination class," *Knowledge-Based Systems*, vol. 204, p. 106185, 2020.

[28] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, "Using collaborative filtering to weave an information tapestry," *Commun. ACM*, vol. 35, p. 61–70, Dec. 1992.

[29] A. Gazdar and L. Hidri, "A new similarity measure for collaborative filtering based recommender systems," *Knowledge-Based Systems*, vol. 188, p. 105058, 2020.

[30] S. Doshi, "Brief on Recommender Systems." https://towardsdatascience.com/brief-on-recommender-systems-b86a1068a4dd, Feb 2019. Accessed: 2020-09-21.

[31] B. Walek and V. Fojtik, "A hybrid recommender system for recommending relevant movies using an expert system," *Expert Systems with Applications*, vol. 158, p. 113452, 2020.

[32] K. M. Gilleade and A. Dix, "Using frustration in the design of adaptive videogames," *ACM International Conference Proceeding Series*, vol. 74, pp. 228–232, 2004.

[33] B. Magerko, "Adaptation in digital games," *Computer*, vol. 41, pp. 87–89, jun 2008.

[34] M. Nitsche, M. Nitsche, C. Ashmore, C. Ashmore, W. Hankinson, W. Hankinson, R. Fitzpatrick, R. Fitzpatrick, J. Kelly, J. Kelly, K. Margenau, and K. Margenau, "Designing procedural game spaces: A case study," *Proceedings of FuturePlay*, pp. 10–12, 2006.

[35] J. Togelius, R. De Nardi, and S. M. Lucas, "Towards automatic personalised content creation for racing games," *Proceedings of the 2007 IEEE Symposium on Computational Intelligence and Games, CIG 2007*, no. Cig, pp. 252–259, 2007.

[36] N. Shaker, G. Yannakakis, and J. Togelius, "Towards automatic personalized content generation for platform games," *Proceedings of the 6th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, AIIDE 2010*, no. Hudlicka 2008, pp. 63–68, 2010.

[37] M. Jennings-Teats, G. Smith, and N. Wardrip-Fruin, "Polymorph: Dynamic Difficulty Adjustment through level generation," *Workshop on Procedural Content Generation in Games, PC Games 2010, Co-located with the 5th International Conference on the Foundations of Digital Games*, 2010.

[38] Remedy Entertainment, "Max payne 3." https://support.rockstargames.com/articles/200152416/Information-about-difficulty-settings-in-Max-Payne-3, 2012. Accessed: 2020-02-10.

[39] B. Magerko, C. Heeter, B. Medler, and J. Fitzgerald, "Intelligent adaptation of digital game-based learning," *ACM Future Play 2008 International Academic Conference on the Future of Game Design and Technology, Future Play: Research, Play, Share*, pp. 200–203, 2008.

[40] S. Kazmi and I. J. Palmer, "Action recognition for support of adaptive gameplay: A case study of a first person shooter," *International Journal of Computer Games Technology*, vol. 2010, 2010.

[41] S. Arnold, J. Fujima, A. Karsten, and H. Simeit, "Adaptive behavior with user modeling and storyboarding in serious games," *Proceedings - 2013 International Conference on Signal-Image Technology and Internet-Based Systems, SITIS 2013*, pp. 345–350, 2013.

[42] Rockstar Games, "Grand theft auto 5." https://www.rockstargames.com/V/, 2013. Accessed: 2020-02-12.

[43] A. Rietveld, S. Bakkes, and D. Roijers, "Circuit-adaptive challenge balancing in racing games," *Conference Proceedings - 2014 IEEE Games, Media, Entertainment Conference, IEEE GEM 2014*, 2015.

[44] S. C. Bakkes, P. H. Spronck, and H. Jaap van den Herik, "Opponent modelling for case-based adaptive game AI," *Entertainment Computing*, vol. 1, no. 1, pp. 27–37, 2009.

[45] Epic Games, "Unreal tournament." https://store.steampowered.com/app/13240/Unreal_Tournament_Game_of_the_Year_Edition/, 2000. Accessed: 2020-02-12.

[46] T. Hartley and Q. Mehdi, "Online action adaptation in interactive computer games," *Computers in Entertainment*, vol. 7, no. 2, pp. 1–31, 2009.

[47] P. Spronck, M. Ponsen, I. Sprinkhuizen-Kuyper, and E. Postma, "Adaptive game AI with dynamic scripting," *Machine Learning*, vol. 63, no. 3, pp. 217–248, 2006.

[48] D. Charles and M. Black, "Dynamic Player Modelling: A Framework for Player-centred Digital Games," *Proceedings of 5th International Conference on Computer Games: Artificial Intelligence, Design and Education (CGAIDE'04)*, vol. Microsoft, no. April, pp. 29–35, 2004.

[49] R. Bartle, "Hearts, Clubs, Diamonds, Spades: Players Who Suit Muds," *Journal of MUD Research*, vol. 1, no. 1, p. 19, 1996.

[50] P. Sweetser and P. Wyeth, "GameFlow," *Computers in Entertainment*, vol. 3, p. 3, jul 2005.

[51] N. Peever, D. Johnson, and J. Gardner, "Personality & video game genre preferences," *ACM International Conference Proceeding Series*, pp. 3–5, 2012.

[52] A. Nagle, P. Wolf, and R. Riener, "Towards a system of customized video game mechanics based on player personality: Relating the Big Five personality traits with difficulty adaptation in a first-person shooter game," *Entertainment Computing*, vol. 13, pp. 10–24, 2016.

[53] G. Andrade, H. Santana, A. Furtado, A. Gouveia, A. Leitao, and G. Ramalho, "Online Adaptation of Computer Games Agents: A Reinforcement Learning Approach," *II Workshop de Jogos e Entretenimento Digital*, pp. 105–112, 2003.

[54] T. Angevine, *MIMICA: A GENERAL FRAMEWORK FOR SELF-LEARNING COMPANION AI BEHAVIOR*. PhD thesis, California Polytechnic State University, San Luis Obispo, California, jun 2016.

[55] Monolith Productions, "Middle-earth: Shadow of war." https://www.shadowofwar.com, 2017. Accessed: 2020-02-14.

[56] C. Wood, "New middle-earth: Shadow of war trailer showcases dynamic orc stories." https://www.playstationlifestyle.net/2017/08/31/middle-earth-shadow-of-war-trailer-showcases-unique-dynamic-orc-stories/, 2017. Accessed: 2020-02-14.

[57] N. Justesen, M. G. Duque, D. C. Jaramillo, J.-B. Mouret, and S. Risi, "Learning a Behavioral Repertoire from Demonstrations," *CoRR*, jul 2019.

[58] O. O. Adelaja, *Dynamic story writer for computer role-playing games*. PhD thesis, Ashesi University, 2019.

[59] Nintendo, "Legend of zelda link's awakening." https://www.zelda.com/links-awakening/gameplay/, 2019. Accessed: 2020-09-16.

[60] W. IJsselsteijn, Y. de Kort, and K. Poels, *The Game Experience Questionnaire*. Technische Universiteit Eindhoven, 2013.

[61] "The state of online gaming – 2020," tech. rep., Limelight Networks, 2020.

Appendix A

Procedural Generation Full Output Example

This appendix contains an example output of the procedural generation of a full play session.

```
Starting floor 1 generation
Dungeon Size: (2, 3)
Shortest path to exit: 5
Dead ends: 1 / 1
   |
 E |
 ↓ | ↓
--------
 ↑ | ↑
  →|←
   | ↓
--------
   | ↑
 S→|←
   |

   |
 E |C,E
   |
--------
   |
 C | C
   |
--------
   |
 S | C
   |

-- Room Distribution --
Combat: 1 (25) || Puzzle: 0 (0) || Corridor: 3 (75) || Total: 4
Finished floor 1 generation
Starting floor 2 generation
Dungeon Size: (4, 2)
Shortest path to exit: 6
Dead ends: 1 / 2
   |   |   |
  →|← →|← →|←
 ↓ | ↓ |   | ↓
----------------
 ↑ | ↑ |   | ↑
 E |   |   | S
   |   |   |

   |   |   |
C,N|C,E| C |P,E
   |   |   |
----------------
   |   |   |
 E |C,H|   | S
   |   |   |

-- Room Distribution --
Combat: 3 (60) || Puzzle: 1 (20) || Corridor: 1 (20) || Total: 5
Finished floor 2 generation
Starting floor 3 generation
Dungeon Size: (2, 4)
Shortest path to exit: 7
Dead ends: 1 / 1
   |
 S→|←
   | ↓
--------
   | ↑
  →|←
 ↓ |
--------
 ↑ |
  →|←
 ↓ | ↓
```

```
--------
 ↑ | ↑
   | E
   |

   |
 S |P,H
   |
--------
   |
C,H| C
   |
--------
   |
 C |P,N
   |
--------
   |
C,E| E
   |

-- Room Distribution --
Combat: 2 (33) || Puzzle: 2 (33) || Corridor: 2 (33) || Total: 6
Finished floor 3 generation
Starting floor 4 generation
Dungeon Size: (4, 3)
Shortest path to exit: 7
Dead ends: 1 / 2
   |   |   |
  →|← →|← →|←S
 ↓ |   |   |
----------------
 ↑ |   |   |
  →|← →|←E |
 ↓ |   |   |
----------------
 ↑ |   |   |
   |   |   |
   |   |   |

   |   |   |
C,N| C |P,H| S
   |   |   |
----------------
   |   |   |
C,N| C | E |
   |   |   |
----------------
   |   |   |
C,N|   |   |
   |   |   |

-- Room Distribution --
Combat: 3 (50) || Puzzle: 1 (17) || Corridor: 2 (33) || Total: 6
Finished floor 4 generation
Starting floor 5 generation
Dungeon Size: (4, 3)
Shortest path to exit: 8
Dead ends: 2 / 4
   |   |   |
   |  →|← →|←
   | ↓ |   | ↓
----------------
   | ↑ |   | ↑
  →|←  | E→|←
   | ↓ |   |
----------------
   | ↑ |   |
 S→|← →|←  |
```

```
   |   |   |
   |   |   |
   |C,N|P,N|C,E
   |   |   |
---------------
   |   |   |
C,H| C | E |P,N
   |   |   |
---------------
   |   |   |
 S | C |C,H|
   |   |   |

-- Room Distribution --
Combat: 4 (50) || Puzzle: 2 (25) || Corridor: 2 (25) || Total: 8
Finished floor 5 generation
Starting floor 6 generation
Dungeon Size: (4, 4)
Shortest path to exit: 13
Shortest path to exit: 13
Shortest path to exit: 13
Dead ends: 2 / 3
   |   |   |
  →|← →|← →|←S
 ↓ |   |   |
---------------
 ↑ | ↑ |   |
   |   | E |
 ↓ | ↓ | ↓ | ↓
---------------
 ↑ |   | ↑ | ↑
   |   |  →|←
 ↓ | ↓ |   | ↓
---------------
 ↑ | ↑ |   | ↑
  →|← →|← →|←
   |   |   |

   |   |   |
P,H| C |C,N| S
   |   |   |
---------------
   |   |   |
 C |   | E |C,H
   |   |   |
---------------
   |   |   |
P,N|C,H|C,N| C
   |   |   |
---------------
   |   |   |
C,H|C,H| C |P,H
   |   |   |

-- Room Distribution --
Combat: 6 (46) || Puzzle: 3 (23) || Corridor: 4 (31) || Total: 13
Finished floor 6 generation
```

Appendix B

User Log Example

This appendix contains an example of the user log generated by the game during gameplay.

	A	B	C	D	E	F	G
1	**Dungeon Floor**	**Dungeon Size X**	**Dungeon Size Y**	**Dungeon Area**	**Shortest Path to Exit**	**Dead Ends Explored**	**Dead Ends Total**
2	1	3	2	6	3	1	1
3	2	3	3	9	8	1	1
4	3	3	2	6	3	0	0
5	4	4	3	12	7	1	2
6	5	3	3	9	8	0	1
7	6	3	4	12	11	1	1
8	7	4	3	12	10	1	1
9	8	4	3	12	10	1	1
10	9	4	4	16	11	0	3
11	10	4	4	16	10	2	2
12	11	4	3	12	9	2	2
13	12	3	4	12	8	0	1
14	TOTAL				98	10	16

	H	I	J	K	L	M	N
1	**Dead Ends Explored %**	**Rooms Explored**	**Room Total**	**Rooms Explored %**	**Rooms Traversed**	**Times Mininum Path Traversed**	**Floor HP Lost**
2	100	4	4	100	7	2.33	0
3	100	9	9	100	10	1.25	4
4	100	3	3	100	3	1	0
5	50	8	11	73	9	1.29	2
6	0	8	9	89	8	1	0
7	100	12	12	100	13	1.18	9
8	100	11	11	100	12	1.2	2
9	100	11	11	100	30	3	6
10	0	11	14	79	11	1	7
11	100	12	12	100	14	1.4	8
12	100	11	11	100	13	1.44	5
13	0	7	11	64	7	0.88	10
14	63	107	118	91	137	1.4	53

	O	P	Q	R	S	T	U
1	**Would die old HP?**	**Died?**	**Quit?**	**Time Spent in Combat**	**Percent Time Spent in Combat**	**Time Spent in Puzzles**	**Percent Time Spent in Puzzles**
2	FALSE	FALSE	FALSE	6.52	17	0	0
3	FALSE	FALSE	FALSE	55.9	58	0	0
4	FALSE	FALSE	FALSE	0	0	0	0
5	FALSE	FALSE	FALSE	13.58	20	32.23	48
6	FALSE	FALSE	FALSE	0	0	6.38	21
7	TRUE	FALSE	FALSE	62.63	44	49.9	35
8	FALSE	FALSE	FALSE	15.7	14	43.65	40
9	TRUE	FALSE	FALSE	58.05	29	27.88	14
10	TRUE	FALSE	FALSE	50.55	52	17.13	18
11	TRUE	FALSE	FALSE	114.66	70	6.9	4
12	FALSE	FALSE	FALSE	43.18	37	28.23	24
13	TRUE	TRUE	FALSE	41.23	38	16.92	16
14				462	39	229.22	19

	V	W	X	Y	Z	AA	AB	AC
1	**Floor Time Taken**	**# Easy Combat**	**# Normal Combat**	**# Hard Combat**	**Total Combat #**	**Combat Room %**	**# Easy Puzzle**	**# Normal Puzzle**
2	38.69	1	0	0	1	50	0	0
3	97.2	2	3	0	5	71	0	0
4	8.03	0	0	0	0	0	0	0
5	66.56	0	3	0	3	33	0	4
6	30.96	0	1	0	1	14	0	1
7	143.85	0	0	3	3	30	0	0
8	109.02	0	0	1	1	11	0	1
9	197.65	0	1	3	4	44	0	0
10	97.2	0	1	5	6	50	0	1
11	163.41	0	2	4	6	60	0	1
12	117.03	0	2	1	3	33	0	0
13	107.54	0	1	4	5	56	0	1
14	1177.14	3	14	21	38	40	0	9

	AD	AE	AF	AG	AH	AI	AJ	AK
1	**# Hard Puzzle**	**Total Puzzle #**	**Puzzle Room %**	**# Corridors**	**Corridor Room %**	**Total Content Rooms**	**Room Content 0x0**	**Room Difficulty 0x0**
2	0	0	0	1	50	2		
3	0	0	0	2	29	7	Combat	Normal
4	0	0	0	1	100	1	Corridor	
5	0	4	44	2	22	9	Combat	Normal
6	0	1	14	5	71	7	Corridor	
7	3	3	30	4	40	10	Corridor	
8	3	4	44	4	44	9	Puzzle	Hard
9	2	2	22	3	33	9	Start	
10	2	3	25	3	25	12	Start	
11	0	1	10	3	30	10	Corridor	
12	3	3	33	3	33	9	Puzzle	Hard
13	1	2	22	2	22	9	Exit	
14	14	23	24	33	35	94		

	AL	AM	AN	AO	AP	AQ
1	**Room HP Lost 0x0**	**Room Solution Time 0x0**	**Room Explored 0x0**	**Room Content 1x0**	**Room Difficulty 1x0**	**Room HP Lost 1x0**
2				Combat	Easy	0
3	1	14.3	TRUE	Combat	Easy	0
4		3.85	TRUE	Exit		
5	0		FALSE	Puzzle	Normal	
6		4.17	TRUE	Exit		
7		2.92	TRUE	Combat	Hard	1
8		10.02	TRUE	Corridor		
9		2.01	TRUE	Corridor		
10		2	TRUE	Combat	Hard	0
11		7.62	TRUE	Corridor		
12		8.97	TRUE	Corridor		
13			FALSE	Combat	Normal	0
14						

	AR	AS	AT	AU	AV	AW
1	**Room Solution Time 1x0**	**Room Explored 1x0**	**Room Content 2x0**	**Room Difficulty 2x0**	**Room HP Lost 2x0**	**Room Solution Time 2x0**
2	6.52	TRUE				
3	8.12	TRUE	Start			3.02
4	1.78	TRUE				
5		FALSE	Combat	Normal	0	
6	1.33	TRUE	Start			2.08
7	21.25	TRUE	Puzzle	Hard		11.37
8	4	TRUE	Puzzle	Hard		15.92
9	9.82	TRUE	Combat	Hard	1	18.63
10		FALSE	Combat	Hard	0	
11	4	TRUE	Combat	Normal	1	15.38
12	5.47	TRUE	Corridor			4.65
13	27.73	TRUE	Puzzle	Hard		10.18
14						

	AX	AY	AZ	BA	BB	BC
1	**Room Explored 2x0**	**Room Content 3x0**	**Room Difficulty 3x0**	**Room HP Lost 3x0**	**Room Solution Time 3x0**	**Room Explored 3x0**
2						
3	TRUE					
4						
5	FALSE	Combat	Normal	2	13.58	TRUE
6	TRUE					
7	TRUE					
8	TRUE	Start			3.12	TRUE
9	TRUE	Exit			1.22	TRUE
10	FALSE					
11	TRUE	Start			2.71	TRUE
12	TRUE	Puzzle	Hard		10.15	TRUE
13	TRUE					
14						

	BD	BE	BF	BG	BH	BI
1	**Room Content 0x1**	**Room Difficulty 0x1**	**Room HP Lost 0x1**	**Room Solution Time 0x1**	**Room Explored 0x1**	**Room Content 1x1**
2	Exit			3.52	TRUE	Corridor
3	Exit			2.67	TRUE	Corridor
4	Start			2.4	TRUE	
5						Exit
6	Puzzle	Normal		6.38	TRUE	Combat
7	Corridor			6.03	TRUE	Combat
8	Corridor			2.5	TRUE	Combat
9	Corridor			5.8	TRUE	Corridor
10	Corridor			2.5	TRUE	Combat
11	Combat	Hard	1	30.73	TRUE	Combat
12	Corridor			8.25	TRUE	Combat
13	Combat	Hard	10		TRUE	Puzzle
14						

	BJ	BK	BL	BM	BN	BO
1	**Room Difficulty 1x1**	**Room HP Lost 1x1**	**Room Solution Time 1x1**	**Room Explored 1x1**	**Room Content 2x1**	**Room Difficulty 2x1**
2			5.13	TRUE	Start	
3			7.77	TRUE	Combat	Normal
4						
5			1.65	TRUE	Puzzle	Normal
6	Normal	0		FALSE	Corridor	
7	Hard	0	17.78	TRUE	Combat	Hard
8	Hard	2	15.7	TRUE		
9			3.22	TRUE	Combat	Hard
10	Hard	3	15.68	TRUE	Puzzle	Normal
11	Hard	2	18.12	TRUE	Combat	Hard
12	Normal	2	12.78	TRUE	Combat	Normal
13	Normal		6.73	TRUE	Combat	Hard
14						

	BP	BQ	BR	BS	BT	BU
1	**Room HP Lost 2x1**	**Room Solution Time 2x1**	**Room Explored 2x1**	**Room Content 3x1**	**Room Difficulty 3x1**	**Room HP Lost 3x1**
2		5.52	TRUE			
3	1	11.9	TRUE			
4						
5		8.75	TRUE	Corridor		
6		4.9	TRUE			
7	8	23.6	TRUE			
8				Exit		
9	4	12.08	TRUE	Puzzle	Hard	
10			TRUE	Combat	Normal	0
11	2	17.53	TRUE			
12	0	13.85	TRUE	Exit		
13	0	13.5	TRUE			
14						

	BV	BW	BX	BY	BZ	CA
1	**Room Solution Time 3x1**	**Room Explored 3x1**	**Room Content 0x2**	**Room Difficulty 0x2**	**Room HP Lost 0x2**	**Room Solution Time 0x2**
2						
3			Combat	Easy	1	6.33
4						
5	4.87	TRUE	Start			3.13
6			Corridor			3.17
7			Puzzle	Hard		13.77
8	2.17	TRUE	Puzzle	Hard		9.17
9	18.03	TRUE	Combat	Normal	0	16.9
10		FALSE	Corridor			4.6
11			Combat	Normal	1	17.58
12	1.55	TRUE	Start			2.72
13			Combat	Hard	0	
14						

	CB	CC	CD	CE	CF	CG
1	**Room Explored 0x2**	**Room Content 1x2**	**Room Difficulty 1x2**	**Room HP Lost 1x2**	**Room Solution Time 1x2**	**Room Explored 1x2**
2						
3	TRUE	Corridor			4.28	TRUE
4						
5	TRUE	Puzzle	Normal		15.4	TRUE
6	TRUE	Corridor			5.27	TRUE
7	TRUE	Exit			1.97	TRUE
8	TRUE	Puzzle	Normal		8.55	TRUE
9	TRUE					
10	TRUE	Combat	Hard	2	20.78	TRUE
11	TRUE	Combat	Hard	1	15.32	TRUE
12	TRUE	Puzzle	Hard		9.12	TRUE
13	FALSE	Corridor				FALSE
14						

	CH	CI	CJ	CK	CL	CM
1	**Room Content 2x2**	**Room Difficulty 2x2**	**Room HP Lost 2x2**	**Room Solution Time 2x2**	**Room Explored 2x2**	**Room Content 3x2**
2						
3	Combat	Normal	1	15.25	TRUE	
4						
5	Puzzle	Normal		8.08	TRUE	Corridor
6	Corridor			3.67	TRUE	
7	Puzzle	Hard		24.77	TRUE	
8	Corridor			5.3	TRUE	Corridor
9	Combat	Hard	1	10.43	TRUE	Puzzle
10	Combat	Hard	2	14.08	TRUE	
11	Exit			1.88	TRUE	
12	Combat	Hard	3	16.55	TRUE	
13	Corridor			2.65	TRUE	
14						

	CN	CO	CP	CQ	CR	CS
1	**Room Difficulty 3x2**	**Room HP Lost 3x2**	**Room Solution Time 3x2**	**Room Explored 3x2**	**Room Content 0x3**	**Room Difficulty 0x3**
2						
3						
4						
5			4.02	TRUE		
6						
7					Start	
8			15.95	TRUE		
9	Hard		9.85	TRUE		
10					Puzzle	Hard
11					Corridor	
12						
13						
14						

	CT	CU	CV	CW	CX	CY
1	**Room HP Lost 0x3**	**Room Solution Time 0x3**	**Room Explored 0x3**	**Room Content 1x3**	**Room Difficulty 1x3**	**Room HP Lost 1x3**
2						
3						
4						
5						
6						
7		2.63	TRUE	Corridor		
8						
9						
10		8.82	TRUE	Corridor		
11		3.12	TRUE	Puzzle	Normal	
12						
13				Combat	Hard	0
14						

	CZ	DA	DB	DC	DD	DE
1	**Room Solution Time 1x3**	**Room Explored 1x3**	**Room Content 2x3**	**Room Difficulty 2x3**	**Room HP Lost 2x3**	**Room Solution Time 2x3**
2						
3						
4						
5						
6						
7	4	TRUE	Corridor			2.98
8						
9						
10	2.98	TRUE	Puzzle	Hard		8.32
11	6.9	TRUE				
12						
13		FALSE	Start			3.65
14						

	DF	DG	DH	DI	DJ	DK
1	**Room Explored 2x3**	**Room Content 3x3**	**Room Difficulty 3x3**	**Room HP Lost 3x3**	**Room Solution Time 3x3**	**Room Explored 3x3**
2						
3						
4						
5						
6						
7	TRUE					
8						
9						
10	TRUE	Exit			3.45	TRUE
11						
12						
13	TRUE					
14						

Appendix C

Form Filled by the Users

This appendix contains the form that the users had to fill in.

Dungeon Escape Questionnaire

This questionnaire aims to gauge the game experience of playing the "Dungeon Escape" prototype game so that the results can be incorporated in my thesis. Player data and answers on this form are completely anonymous and the collected data will only be used for the thesis. The questionnaire should take about 10 minutes.

This questionnaire should be filled after playing the game "Dungeon Escape", which can be played here: https://ezspecial.itch.io/dungeon-escape

Don't forget to email the files mentioned in the game's page! This questionnaire is invalid without those files.

* Required

General Player Info

1. Play session ID *

The game generates a play session ID for each attempt at the game, your answers should reflect the specific play session that you set here.

_____

2. What is your age group? *

Mark only one oval.

- ◯ <18
- ◯ 18-35
- ◯ 36-49
- ◯ 50+

3. Do you play video games regularly (at least 6h per week) *

Mark only one oval.

- ◯ Yes
- ◯ No

4. Have you played games similar to this one? *

Mark only one oval.

- Yes
- No

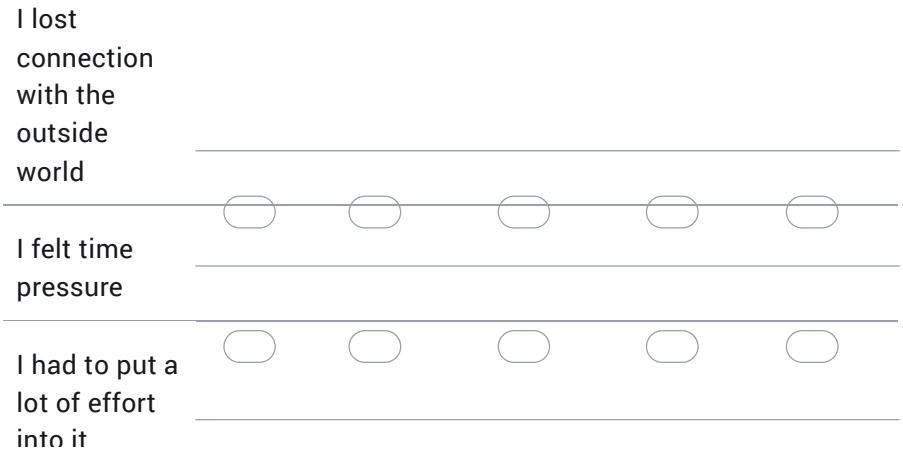
Game Experience Questionnaire - Core

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

Mark only one oval per row.

	Not at all	Slightly	Moderately	Fairly	Extremely
I felt content					
I felt skilful					
I was interested in the game's story					
I thought it was fun					
I was fully occupied with the game					
I felt happy					
It gave me a bad mood					
I thought about other things					
I found it tiresome					
I felt competent					
I thought it was hard					
It was aesthetically pleasing					
I forgot everything around me					
I felt good					

I was good at it					
I felt bored					
I felt successful					
I felt imaginative					
I felt that I could explore things					
I enjoyed it					
I was fast at reaching the game's targets					
I felt annoyed					
I felt pressured					
I felt irritable					
I lost track of time					
I felt challenged					
I found it impressive					
I was deeply concentrated in the game					
I felt frustrated					
It felt like a rich experience					
I lost					



Game Experience Questionnaire - Post-Game

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

Mark only one oval per row.

	Not at all	Slightly	Moderately	Fairly	Extremely
I felt revived					
I felt bad					
I found it hard to get back to reality					
I felt guilty					
It felt like a victory					
I found it a waste of time					
I felt energised					
I felt satisfied					
I felt disoriented					
I felt exhausted					
I felt that I could have done more useful things					
I felt powerful					
I felt weary					
I felt regret					
I felt					

I felt ashamed					
I felt proud					
I had a sense that I had returned from a journey					

Thoughts

7. If you have any suggestions or opinions you can write them here

Congratulations!

Thank you for filling in the questionnaire! It will be a great help for my work. Don't forget to click the Submit button!