MCAST

Creating Difficulty Levels with Reinforcement Learning in a Strategy Game

Georg Grech

Supervisor: David Deguara

June - 2023

A dissertation submitted to the Institute of Information and Communication Technology in partial fulfilment of the requirements for the degree of BSc (Hons) in Multimedia Software Development

Authorship Statement

This dissertation is based on the results of research carried out by myself, is my own composition, and has not been previously presented for any other certified or uncertified qualification.

The research was carried out under the supervision of Mr. David Deguara

3rd June 2023

....

Date Signature

Copyright Statement

In submitting this dissertation to the MCAST Institute of Information and Communication Technology, I understand that I am giving permission for it to be made available for use in accordance with the regulations of MCAST and the Library and Learning Resource Centre. I accept that my dissertation may be made publicly available at MCAST's discretion.

3rd June 2023

. . . . . .

Date

. . . . . .

Signature

Acknowledgements

First and foremost, I would like to thank my mentor, Mr. David Deguara, without whose guidance this research would not have been possible.

I would also like to thank my family, who were a constant source of support and who put up with me when my work put me in the worst of spirits.

Furthermore, I would like to thank my friends, all of whom sought to see me succeed. In particular, I would like to thank my close friends Reece P. and Octavian G., without whom I am not sure how I would have gotten through these past five years. I would also like to thank Matthew P., whose helping hand and our enlightening discussions did not go unappreciated.

Abstract

In video games, difficulty is a vital aspect in forming an enjoyable player experience. Since video game players possess different levels of skill, developers should create difficulty settings that can create an enjoyable and engaging experience for all of them. This study proposes the creation of difficulty levels by modifying the intelligence of an Artificial Intelligence (AI) opponent. This is attempted in a custom-made strategy game using Reinforcement Learning (RL) and Deep Reinforcement Learning (DRL) technologies. During training, versions of an RL agent are periodically extracted and later assigned to in-game difficulty levels. This creates an agent that changes its intelligence to provide a suitable challenge based on the selected difficulty. Testing this implementation with multiple participants, a level of distinction could be observed between difficulty levels, and many participants claimed to find a level of difficulty that served to adequately challenge them.

Keywords: Reinforcement Learning, Deep Reinforcement Learning, Machine Learning, Game Difficulty, Player Experience.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AI	Artificial Intelligence
FSM	Finite State Machine
RL	Reinforcement Learning
MDP	Markov Decision Process
DL	Deep Learning
DRL	Deep Reinforcement Learning
DDA	Dynamic Difficulty Adjustment

Chapter 1: Introduction

1.1 Background

The enjoyment of a task is directly correlated with its level of difficulty; too hard a difficulty causing anxiety, too easy causing boredom. [1]. This extends also to video games, where developers typically try to ensure that the game maintains an enjoyable level of difficulty. However, no single level of difficulty can cater to every type of player, as they can have different backgrounds and experiences playing video games, and so have different levels of skill. As such, developers must ensure that the game remains enjoyable for all players, and in consequence usually create multiple difficulty levels, such as the classic "Easy", "Medium", and "Hard".

The challenge presented by a game can be accomplished through the use of in-game opponents. These opponents, or "agents", as they may be referred to, have became a prominent feature in several games. Much of the research in the field of game development relates to studying Artificial Intelligence (AI) to make more intelligent and/or believable non-playable characters or opponents [4]. Many systems have been adopted and proven effective over the years, such as the Finite State Machine (FSM) [5]. In more recent years, implementations have seen the use of Reinforcement Learning (RL) [3] and Deep Reinforcement Learning (DRL) [6] technologies to create these agents. These RL and DRL-based agents have seen staggering successes even in quite complex games. One such milestone

was OpenAI Five in *Dota 2*, managing to beat the world champion team in a professional setting [7].

There has been thought to use these technologies to achieve difficulty settings, modifying the intelligence of the agent to create an appropriate level of challenge. One such approach used versions of an agent extracted at periodic intervals during training to create levels of difficulty, with promising results [8]. It remains to be seen, however, whether such an approach can work on a more complex strategy game with a heavy emphasis on procedurally generated environments.

1.2 Purpose Statement

The purpose of this research study is to improve player experience via difficulty settings that modify an in-game opponent in a strategy game with a procedurally generated map. By training an agent with Reinforcement Learning techniques and saving versions of it trained to different amounts, distinct difficulty settings can be created and serve to be enjoyable for players. This study aims to analyse the independent variable of the difficulty setting and how it affects dependent variables such as the agent's score to quantitatively assess agent performance, as well as the opinions of participants regarding the level of challenge presented to them.

1.3 Hypothesis and Research Questions

This research presents the following hypothesis: An agent acting as an opponent in a procedurally generated strategy game can be trained to effectively play the

game, following which, three stages of training can be carefully selected and assigned to three difficulty levels: Easy, Medium, and Hard.

From this hypothesis, the following research questions can be identified and will be explored throughout the course of this study.

- What training setup can a reinforcement learning agent use to learn how to effectively play a procedurally generated strategy game?
- How can versions of an agent to be used in different difficulty settings be periodically saved during training?
- How effective are the generated difficulty levels for a positive player experience?

1.4 Outline of Research

This research is divided into several chapters that serve to outline the entire process, allowing a clear view on how research problems are identified and in what manner they are tackled.

In the literature review section, prior studies detailing subjects that relate to this study are explored to allow an understanding of technologies that are to be used in the methodology and determine the current state-of-the-art. These subjects include Player Experience and difficulty, Artificial Intelligence, Reinforcement Learning, Deep Reinforcement Learning, and Dynamic Difficulty Adjustment.

The methodology section documents the process taken to gather and analyse findings that are used to achieve the aim of this study. The development of a

prototype is described in technical detail, from development of the base game, development of the agent's logic, training, and model selection. The manner of data collection, both quantitative and qualitative, is also described, as well how data will be evaluated.

In the findings and discussion section, data gathered from testing is presented, including in-game statistics from both the players and the agent, as well as opinions from participants detailing their observations during gameplay. This data is analysed and discussed in relation to the presented hypothesis and research questions.

In the conclusions section the research process is reiterated by a brief summary and is followed by a presentation of the findings. Limitations of the study are discussed and recommendations for future work are drawn up that have the potential of building further upon this study.

Chapter 2: Literature Review

2.1 Overview

This section explores and discusses past literature concerning the technical details of the subjects and technologies relevant to this study. Furthermore, the methodologies and results of similar studies are reviewed to help identify the ideal manner in which this study should execute its own methodology.

2.2 Player Experience and Game Balance

In video game development, a critical aspect to the success of a video game is a carefully developed and well put-together Player Experience. Player Experience is the branch of User Experience relating to video games [9], while User Experience is commonly defined as how users interact with a piece of technology, and how they are, in turn, affected emotionally [10]. In video games, developers attempt to shape a positive Player Experience mainly through the method of playtesting, refining in-game systems and features to make the experience as enjoyable as it can be [11]. There are multiple facets that can affect the player experience. A critical one is difficulty. The perfect level of difficulty will keep players engaged and ensure that they will not feel anxious, a result of excessive difficulty, or boredom, the usual by-product of a task of insufficient difficulty [1]. Therefore, feedback from playtesting is also used in video game development to refine difficulty until the game considered optimally balanced [12].

The difficulty of a task, however, can be perceived differently between people depending on their level of skill [1]. Therefore, for a task to always maintain a perfect level of difficulty, the challenge presented by a task must increase or decrease relative to the player's skill. This was described in Csikszentmihalyi's "Flow Model" shown in Figure 2.1, with the relationship between player skill and the task's challenge to maintain the perfect level of difficulty being known as the "Flow Channel" [1]. In an attempt to cater for this, some developers opt to create multiple difficulty settings, for instance the traditional "Easy", "Medium", and "Hard" settings. This, however, does not account for any changes in an individual player's skill level, which potentially increases as the player learns the game mechanics, making a previously challenging difficulty setting become too easy [13].

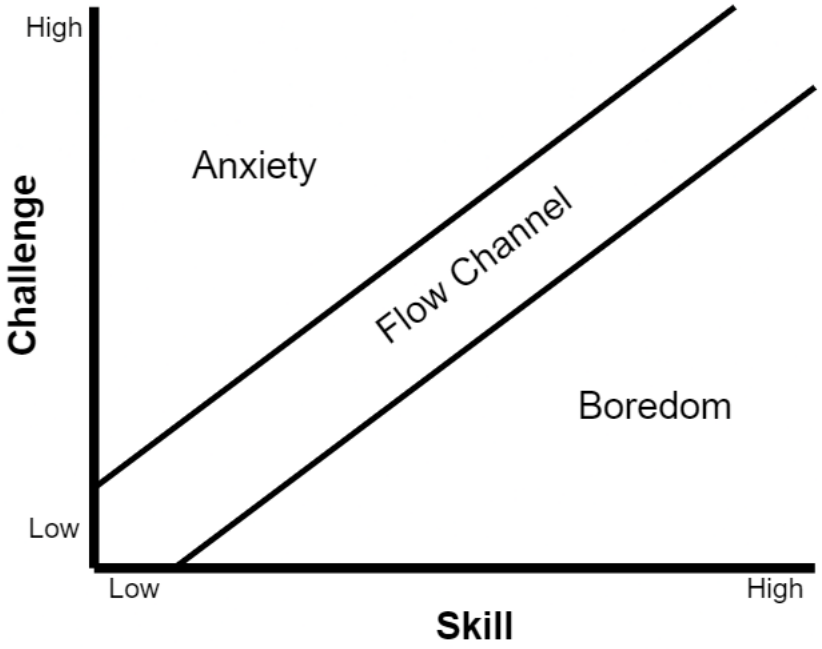

Figure 2.1: *Csikszentmihalyi's flow model diagram [1]*

2.3 Artificial Intelligence in Games

The use of Artificial Intelligence (AI) has become a prominent feature in games [14]. Well-designed implementations can serve to enhance player experience, and in turn add to the game's commercial value [4]. Most popularly, this implementation is done in the form of non-playable characters and opponents. These AI game opponents are developed to play games with two core objectives in mind: to play well, focusing on giving the player a challenge, and/or to play believably, with the AI emulating a human player to provide a more believable and immersive experience [4].

2.3.1 Finite State Machines

The complex and dynamic nature of video games requires that intelligent in-game opponents, often referred to as "agents", be made to take decisions [15]. After developers decide what actions an agent can decide to carry out, it is imperative that a process is developed that allows the agent to make appropriate decisions. Rabin [5] describes the Finite State Machine (FSM) as the most common software pattern that addresses this problem. FSMs describe a relationship between several states, transitions between states that are described by a condition in order to elicit a state change, and actions that are followed within each state [16], Figure 2.2 showing an example of this.

FSMs are simple to implement and debug on smaller scales. However, they can become increasingly difficult to understand and debug as more states and conditions are added [17]. For more dynamic decision-making, their inflexibility

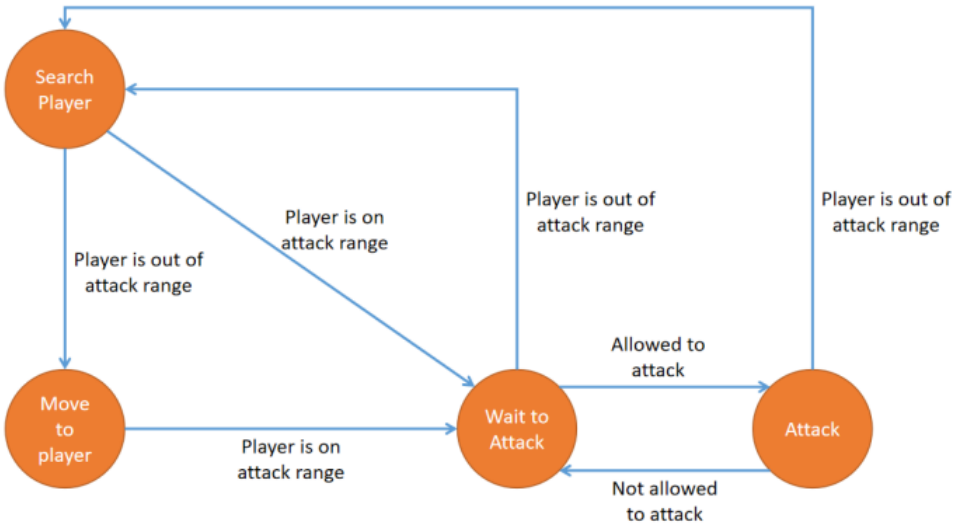

Figure 2.2: *An example of an FSM used by an enemy soldier in the game Khalid ibn Al-Walid [2]*

makes them unfavourable, and they depict predictable behaviour which players tend to be able to figure out [18].

2.4 Machine Learning

2.4.1 Reinforcement Learning

Reinforcement Learning (RL) is a branch of Machine Learning that takes much of its inspiration from biological learning systems. As such, it most closely emulates the learning process of humans and animals out of all machine learning methods [3]. RL concerns itself with how intelligent agents learn from interacting with the environment around them [3]. The most common solution RL systems use to solve this problem is the Markov Decision Process (MDP) [3]. The most important elements of MDPs are states, actions, and rewards [19]. On each step of interaction an MDP-based RL agent is fed an input through which it observes the state of its environment [20]. While in a state, there are sev-

eral actions available to an agent. Unlike an FSM, however, these actions aren't taken when certain conditions are satisfied. Instead, the agent is free to attempt any action available. [20]. Following the action taken, the agent is rewarded or penalised, through which the agent learns over multiple tries the optimal actions to take within a state [19]. After the reward is given, the agent is notified about the new state of the environment, and the cycle repeats [4]. Thus, through this repeated process of training, an RL agent becomes gradually more intelligent, as it prioritizes taking the decisions that maximize its reward [3].

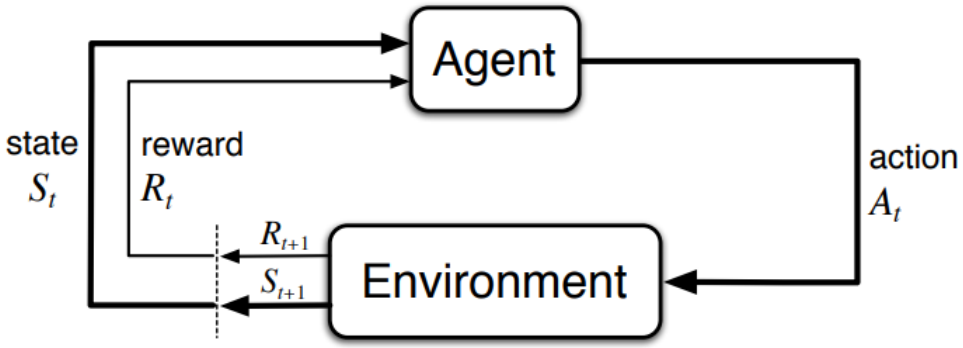

Figure 2.3: *The agent–environment interaction in reinforcement learning. [3]*

2.4.2 Deep Learning

Deep Learning (DL) is another method of Machine Learning that is based on the use of Neural Networks [21]. Neural Networks are systems that are modeled on the structure and functioning of biological brains such as those of humans and animals. In these systems, an artificial neuron receives information processed by other neurons before it, processes it further, and passes it on the the next neuron to continue processing [22]. This layered approach continues until the system ends up with a single output [22]. DL, therefore, can learn from an existing

unprocessed data set [23]. DL algorithms process this raw data using multiple processing layers to find structures and patterns [24]. This process is automatic, not requiring any sort of prior data processing to find and extract features [24].

2.4.3 Deep Reinforcement Learning

Deep Reinforcement Learning (DRL) is a method of Machine Learning that combines RL and DL. DRL uses the DL approach to process a large quantity of raw, complex data and break it down into simpler data to be used by the RL agent [6]. This processing done by a Neural Network allows the RL agent to learn directly from a complex input such as a visual signal composed of pixels [25].

2.4.4 Deep Reinforcement Learning in Games

Agents using DRL have had impressive results in video games. A demonstration of the performance and range a DRL agent can achieve was seen in a 2013 study where a DRL agent was trained to play several Atari 2600 games, using the raw on-screen pixels as observations [26]. In the games of *Breakout*, *Enduro*, and *Pong*, this agent managed to score higher on average than an expert human player [26].

This level of performance have made DRL agents a logical choice to serve as opponents against human players in more competitive games. This includes real-time fighting games such as *Super Smash Bros. Melee* [27] and *Blade & Soul* [28]. In scenarios such as these, where two players directly interact with each other, a technique known as "self-play" may be adopted [29]. With self-play, an agent

learns by playing against past versions of itself, trying to achieve an improved performance [29]. Using this technique, both fighting game agents were able to achieve favourable results against competitive players [27, 28]. However, it is unlikely that a DRL agent will learn how to combat all possible player strategies during training, and therefore may be easily defeated with certain unconventional strategies, as was observed by [27].

DRL has now been tried in competitive strategy games. One of the most notable examples in recent years is OpenAI Five in *Dota 2*. OpenAI Five was also trained using a self-play technique. After the completion of training, it defeated the world champion team at an Esports game [7]. Despite this success, there is the significant drawback of training time required for an agent to reach this level of intelligence playing complex strategy games. For instance, *Dota 2*'s level of complexity required OpenAI Five to observe around 16,000 values, with between 8,000 and 80,000 actions available to the agent on every timestep, resulting in 180 days of training time, spread out over a total of 10 months [7].

2.4.5 Unity ML-Agents

Unity is an immensely popular game engine, used by amateur and professional developers alike [30]. Its popularity has led many Unity developers to attempt to implement modern technologies, with some of these implementations being published as plugins and packages to aid other developers [31]. Machine Learning, including RL and DRL, is one such technology developers have attempted to implement. The most popular package for this is the open-source Unity ML-Agents package, initially developed and published in-house by Unity Technologies [32].

ML-Agents was developed to be as versatile as possible, allowing implementations of DRL in completely different types of scenarios. The package allows support for environments with a single agent or multiple, these agents being able to receive both visual or vector observations, and taking either discrete or continuous actions [32].

The Unity ML-Agents package has been used successfully for the development of intelligent agents as opponents. A 2021 study, for instance, used the package to develop an AI opponent in a fighting game [33]. This agent was trained by observing health and stamina points of itself and its opponent, and being rewarded for actions that get it closer to winning [33]. Another study using Unity ML-Agents sees an agent being trained to play the popular board game *Connect-4* [34]. This agent was successful in learning strategies for effective play, managing to win against a human player [34].

2.5 Dynamic Difficulty Adjustment

A solution that game developers have opted to use in order to make video game difficulty conform to Csikszentmihalyi's Flow Model [1] and cater to players of differing skill levels is that of Dynamic Difficulty Adjustment (DDA). DDA is the process of making automatic adjustments to a video game in real-time to modify its level of difficulty, based on the player's abilities [35]. With these adjustments, an optimal level of difficulty is attempted to be found for every individual player regardless of how their skills fluctuate, keeping them in the Flow Channel and negating boredom or frustration [36].

Many players have been shown to enjoy DDA over static difficulty settings. A 2017 study, for instance, tested a dynamically adjusted Match-3 game on a group of participants. These participants reported a higher level of engagement compared to a focus group that played a version of the same game with a static difficulty [37]. Participants using DDA willingly played more rounds of the game and had an increased gameplay duration [37].

Players, however, have been shown to enjoy a good challenge more than winning, therefore DDA systems should be careful not to exaggerate when lowering difficulty. This was pointed out by [38], who carried out an evaluation between five versions of an AI opponent in a real-time strategy game, two of which static, three of which using dynamic difficulty. In this study, two dynamic versions of the opponent were ranked as the most enjoyable by players. While the static versions did not rank as high, they were higher than the third dynamic version, which had a special condition that lowered its difficulty in a manner that always let the player win if they were seriously struggling [38]. This exaggerated lowering of difficulty made the opponent feel boring, even if its DDA system was very similar to one of the high-ranking dynamic opponents [38].

2.5.1 Determining Player Skill

To develop an effective DDA system, developers must take care to choose the variables that best determine the player's skill level, although these can vary greatly between games. In Hunicke's implementation in a game developed in the *Half-Life* engine, the Hamlet DDA system was used to monitor the game's core inventory, including health, shielding, ammunition, and weapons, with adjust-

ments in the level being carried out partly based on fluctuations of items in the inventory [12]. When in a combat encounter with an enemy, the DDA system viewed rapid decreases of items in the player's inventory as a sign of struggle, and therefore the system intervened to carry out necessary level modifications to aid the player [12].

2.5.2 Player Skill in Strategy games

In competitive strategy games, it may be difficult to determine player performance by exclusively analysing the player's variables. Since players compete with an opponent in these types of games, comparing player and opponent variables may give a clearer picture of how the player is performing. In the implementation by [38] in a real-time strategy game, for instance, both the player and enemy control multiple units, each with its own health value. In this implementation, an algorithm compares the number of units the player and enemy each control and their health points, and uses this data to calculate who between the player and the enemy is in advantage and to what degree, and with this information carries out the modifications required for balancing [38]. Alternatively, in competitive strategy games, both the player and enemy usually have a set of objectives to accomplish, typically attempting to accomplish them quicker than their opponent. Therefore, performance of a player can be measured through the rate of progress. This approach can be observed in an implementation of DDA in the game *Dota 2* by [39]. In *Dota 2*, two teams own a number of towers each, and one of their main objectives is to destroy the other team's towers. [39]'s DDA system kept track of the number of the enemy team's towers that were destroyed by the

player's team, indicating their overall progress. If this number was increasing too rapidly, this signified a potential unbalance, thus the game increased its difficulty to keep at a considerable level of challenge [39].

2.5.3 Game Modification

After developers decide on the ideal parameters for measuring skill level, it is then to determined what changes will be carried out in-game to accordingly adjust difficulty. A manner this can be accomplished is through the supply of helpful in-game items, an example of which can be seen in Hunicke's implementation, where in response to the system detecting the player struggling, a health pack may spawn somewhere within the scene, as well as health packs and ammunition having a higher chance of being dropped by an eliminated enemy [40]. Modification of in-game opponents can also prove effective as can be seen in Chowdhury and Katchabaw's study [41]. In their implementation of DDA in a variant of Pac-Man, adjustments are made to the attributes of enemies, modifying their movement speed and area of vision, among others.

2.5.4 Applications of RL for DDA

There has been thought to combine RL and DRL methods with DDA, modifying agent intelligence to achieve an enjoyable game balance. One such study attempted this in a real-time fighting game [42]. This study initially trained an RL agent to the best of its ability by playing against another AI agent with random behaviour. Outside of training, this agent modified its behaviour based on the life difference of itself and its opponent, whenever necessary switching to another per-

formance level [42]. These performance levels were based on the knowledge of the optimal actions the agent learned during training. Whenever the agent was required to take an action, it had all of the game's thirteen actions ranked best to worst to take in that specific state according to its knowledge. Therefore, the agent had thirteen performance levels it could switch to. The best performance level saw the agent always choosing the best option, the second-best level choosing the second-best option, and so on, with the last level choosing the worst of all the possible actions [42]. When playing against opponents using different AI systems, such as random behaviour, FSM, and traditional RL, the adaptive RL agent consistently played with an equal skill level to its opponent [42].

A simpler approach that avoided directly accessing the agent's knowledge was adopted by [8] when developing an opponent for the first-person shooter game *Unreal Tournament 2004*. This study trained an RL agent by having it play against a bot using the hardest native difficulty setting. During training, milestones of the agent's knowledge were periodically saved and stored, therefore the final agent's performance could be modified by changing its knowledge to a version at a different point in training [8]. This was used to develop a DDA system based on the difference of kills and deaths during a game. If it was detected that this difference went outside a certain range, the agent was modified to use the previous or the next milestone, depending on whether the difference was positive or negative [8]. This system proved effective against the native bots, with the agent achieving a very balanced win/loss ratio in each pre-programmed difficulty setting [8]. Neither of these two studies, however, tested their implementations on

human players [8, 42].

Chapter 3: Research Methodology

3.1 Research Pipeline

Figure 3.1 outlines this study's research methodology, divided into three major segments. Setup consisted of deciding on the major technologies with which to develop the agents, settling on Unity ML-Agents, as well as gather other necessary assets for developing the prototype. The next stage saw the development of the game's core functionalities, followed by development of the agent and training, as well as the difficulty levels and the functionality for data collection. Finally the prototype was evaluated by the participants, from which was extracted quantitative data from gameplay and performance metrics, as well as qualitative data from the participants.

3.2 Prerequisites and Tools

Before starting development on the prototype, the ML-Agents project was cloned off GitHub[1] into a local directory. The version used in this project was the latest version at the time, Release 19. With this project also came included TensorBoard, a tool for analysing statistics during training. The prototype described in this methodology was created using the Unity game engine version 2021.1.22f1, and its project was located inside the previously cloned repository. Using the Unity Package Manager, the ML Agents and ML Agents Extensions packages

[1] `https://github.com/Unity-Technologies/ml-agents.git`

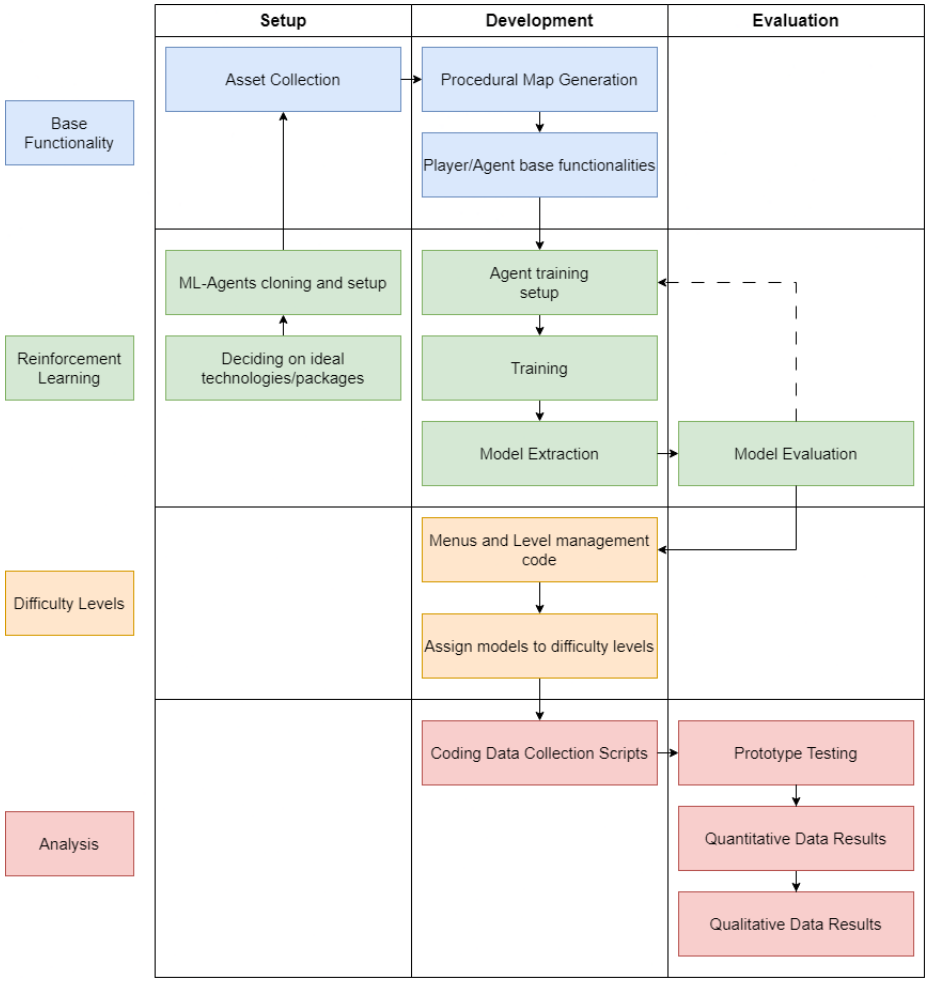

Figure 3.1: *Pipeline followed by the methodology*

were loaded locally from the ML-Agents repository. Additionally, the NavMesh-Components package was downloaded from GitHub[2] and loaded into the project.

3.3 Prototype Game Outline

A prototype game was developed to implement and test Reinforcement Learning for this research. The core gameplay consists of traversing a map in a top-down setting, filling the player's inventory by gathering resources and exchanging them at their base for points. Resources on the map are spawned procedurally with

[2] `https://github.com/Unity-Technologies/NavMeshComponents.git`

random positioning and being of a random type. The game also procedurally spawns obstacles on the map to provide the player a challenge in getting near a potentially valuable resource.

The main challenge presented to the player is through an AI opponent which could take the same actions as the player. The objective for both the human player and the AI opponent is to gather as many points as possible before a timer of five minutes runs out. The game ends when the timer finishes, the player winning if they manage to gather more than the AI opponent, otherwise resulting in a loss if the opponent has a greater score or a draw if both are equal.

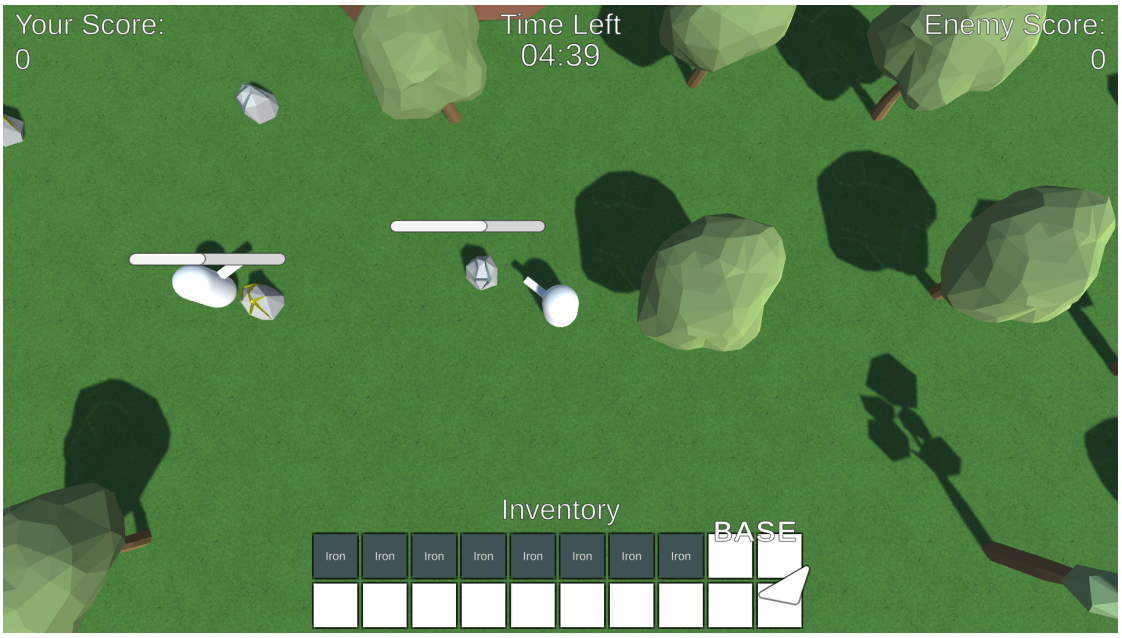

Figure 3.2: *Screenshot of gameplay showing the player and opponent gathering resources*

For the purpose of this research in analysing difficulty, the difficulty setting could be chosen via the main menu before starting a game. Players are given a choice between three difficulty levels: Easy, Medium, and Hard. This setting modifies the agent by switching between versions with different training durations,

reminiscent to [8].

3.3.1 Resources

There are three types of resources that can spawn in the game. These are Wood, Iron, and Gold. The spawned Resource objects contain an amount of items which, after interaction, one-by-one fill the player or agent's inventory until the resource depletes or their inventory is full. This process is illustrated in Figure 3.3.

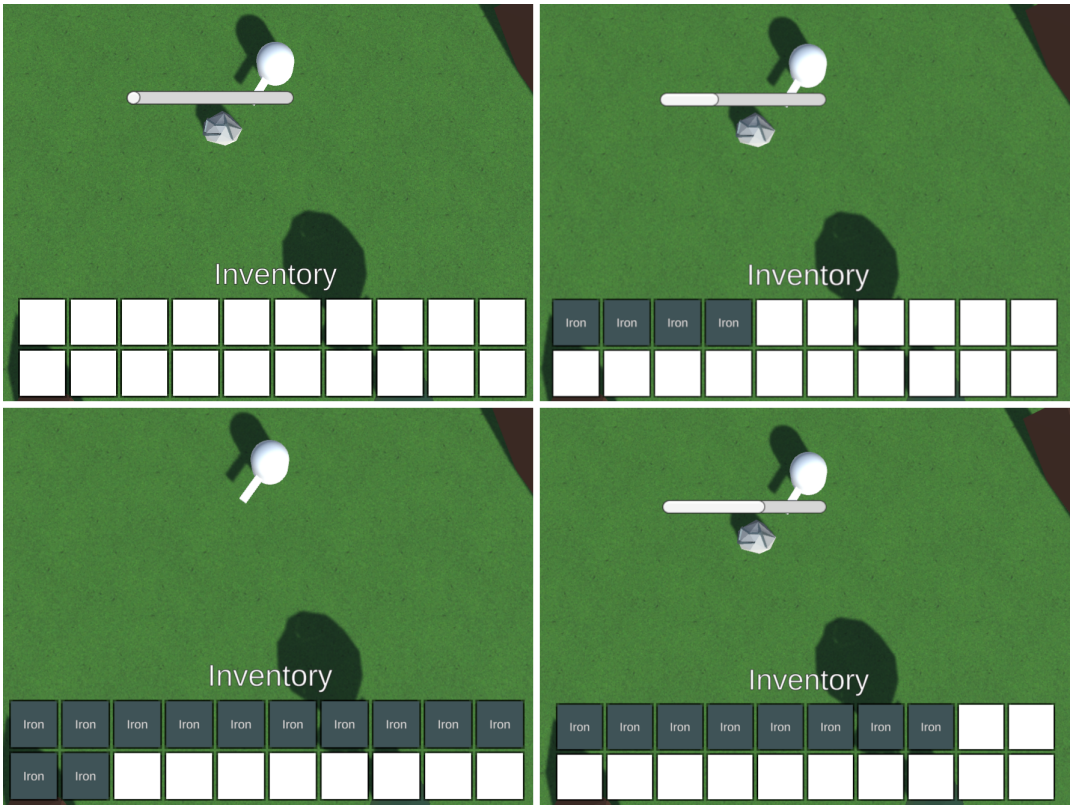

Figure 3.3: *Clockwise from top-left, the player gathering a resource and filling their inventory*

Resource objects of different types are visually distinct, as can be seen in Figure 3.4. They also have different chances of spawning, as listed in Table 3.1.

Resource types have distinct differences in time required to gather, inventory space taken, and worth in points. Per inventory item, the resource types have the

Resource Type	Percentage Chance to Spawn
Wood	**75%**
Iron	**20%**
Gold	**5%**

Table 3.1: *Resource type spawn chances*

attributes seen in Table 3.2.

Resource Type	Score	Gather Time	Inventory Space Taken
Wood	**5**	**1.5**	**1**
Iron	**15**	**3**	**4**
Gold	**30**	**5**	**5**

Table 3.2: *Resource attributes per inventory item*

The resource objects, taking account all of their contained items, have the attributes listed in Table 3.3.

Resource	Drop Amount	Total Score	Total Gather Time	Total Inventory
Wood	**4**	**20**	**6**	**4**
Iron	**3**	**45**	**9**	**12**
Gold	**2**	**60**	**10**	**10**

Table 3.3: *Resource attributes per spawned object*

3.4 AI Opponent

The main goal when developing the AI opponent was that it should play the game intelligently to provide a challenge to the player. Therefore, it should have some level of understanding of when best to gather a resource, the optimal resource to gather, and when best to return to base.

Figure 3.4: *Types of resources found in the game, as shown by the in-game tutorial panel*

The functionalities of the opponent were created using several AI-related packages. Unity ML-Agents was used to implement DRL for decision making. Unity NavMesh and NavMesh components were used for pathfinding and distance calculation when taking decisions.

3.4.1 Agent Observations

To understand its environment, the agent is supplied with variables to observe via the Unity ML-Agents CollectObservations method, seen in Figure 3.5. All of the numerical observations are normalised between 0 and 1 . Variables observed are:

- Game Time left

  The game duration left, normalised by dividing the seconds left with the maximum game seconds.

- Agent Inventory

  Inventory available to the agent, normalised by dividing by the maximum inventory size.

The agent also observes variables related to a selected resource. These are:

- Normalised Distance from Agent
- Normalised Distance from Base
- Normalised Inventory space taken by resource
- Resource Type

The process of resource selection and how these variables are normalised will be described later on.

```
public override void CollectObservations(VectorSensor sensor)
{
    try
    {
        if (targetTransform != null)
        {
            sensor.AddObservation(targetPDistanceNormalised);
            sensor.AddObservation(targetBDistanceNormalised);
            sensor.AddOneHotObservation((int)targetType, numOfTypes);
            sensor.AddObservation(targetInvAmountLeftNormalised);

        }
        sensor.AddObservation((float)enemyPlayer.inventoryAmountFree / enemyPlayer.maxInventorySize);
        sensor.AddObservation((float)enemyPlayer.gameManager.timerSecondsLeft / enemyPlayer.gameManager.timerTotalSeconds);
    }

    catch
    {
        Debug.Log("Exception caught in observations");
    }
}
```

Figure 3.5: *The agent's CollectObservations method*

3.4.2 Agent Actions

The agent's logic was based off two main actions, GatherResource and Return-ToBase. Both actions are similar in functionality. The process begins by giving the agent a destination which it will attempt to move towards. The process then suspends until the destination has been reached. Once it has been detected that the destination has been reached, the agent proceeds to interact with the object

at its destination. The process once again waits until the interaction has concluded, after which the agent decides on the next action to take. This process is illustrated in Figure 3.6

The main difference between the GatherResource and ReturnToBase actions are the destinations. The destination in GatherResource is a resource that the agent has decided it wants to attempt to gather, while in ReturnToBase the destination is the enemy's home base, where the agent attempts to deposit the resources in its inventory.

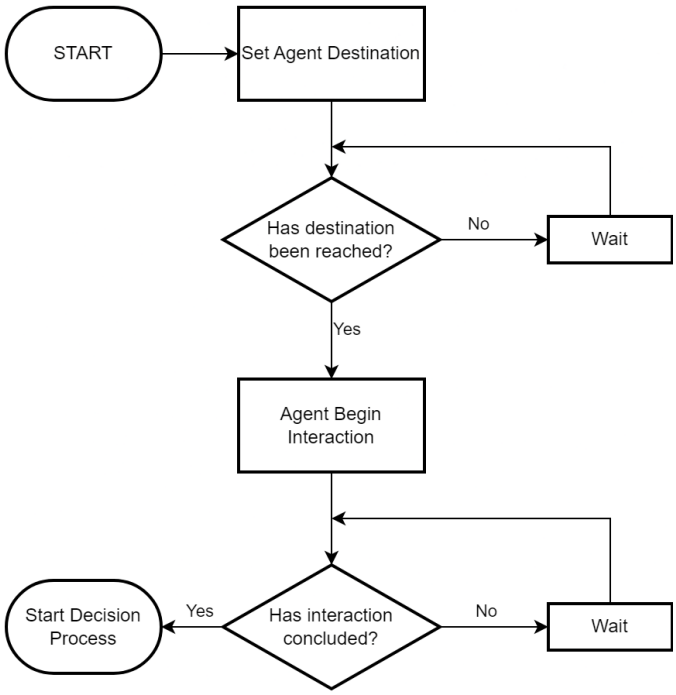

Figure 3.6: *Flowchart of process taken by GatherResource and ReturnToBase actions*

3.4.3 Decision Process

Using features provided by the Unity ML-Agents package, a process was built to that allows the agent to observe the attributes of nearby resources and decide between the previously described actions, either returning to base or selecting a

resource and gathering it. The entire process is described below and visualised in Figure 3.7.

1. **Selecting nearby resources**

   To reduce computational power when gathering information about resources, the agent is limited to only selecting from resources in the immediate vicinity. This limiting was achieved by using a Unity OverlapSphere, saving the colliders of resources within a set range of the agent. If no resources are found within range, the size of the OverlapSphere is increased.

2. **Resource attribute collection**

   Once the agent finds resources in the vicinity, it goes through them and saves their attributes. This includes the two distance values: how distant the resource is from the agent, and how distant the resource is from the base. To take into account the procedurally generated obstacles, NavMesh distance was used, giving a more accurate indication of the length of the path the agent would have to travel. The furthest and closest of these distances are saved to be used for observation normalisation later on. The type of resource is also saved, along with the inventory space gathering this particular resource would take.

3. **Updating Observations**

   The next step in this process is to update the agent with the appropriate observations. Before setting the observations, attributes need to be nor-

malised between 0 and 1. In the case of distances, this is achieved with the following equation:

$$ND = \frac{ResourceDistance - MinDistance}{MaxDistance - MinDistance} \tag{3.1}$$

The resulting normalised distance ND may never be less than 0 or greater than 1.

$$0 \leq ND \leq 1 \tag{3.2}$$

$ND = 0$ indicates that this resource is the closest, while $ND = 1$ indicates that it is the furthest of the scanned resources.

Amount of inventory taken by the resource is normalised using the agent's maximum inventory size. The agent's observations are finally updated with the normalised attributes.

4. **Requesting Decision**

After updating the agent's observations, the agent calls the Unity ML-Agents method RequestDecision to let the agent decide between three discrete actions it can take:

- Gather this resource

  The agent begins the GatherResource action, setting its destination to the resource currently being observed and attempts to gather it.

- Return to base

The agent begins the ReturnToBase action, setting its destination to its base and attempts to deposit what it has collected till that point.

- Observe next resource

  The agent iterates through the next resource in the list and repeats the steps of updating the observations and calling RequestDecision. If the end of the list has been reached, the agent starts going through the list anew.

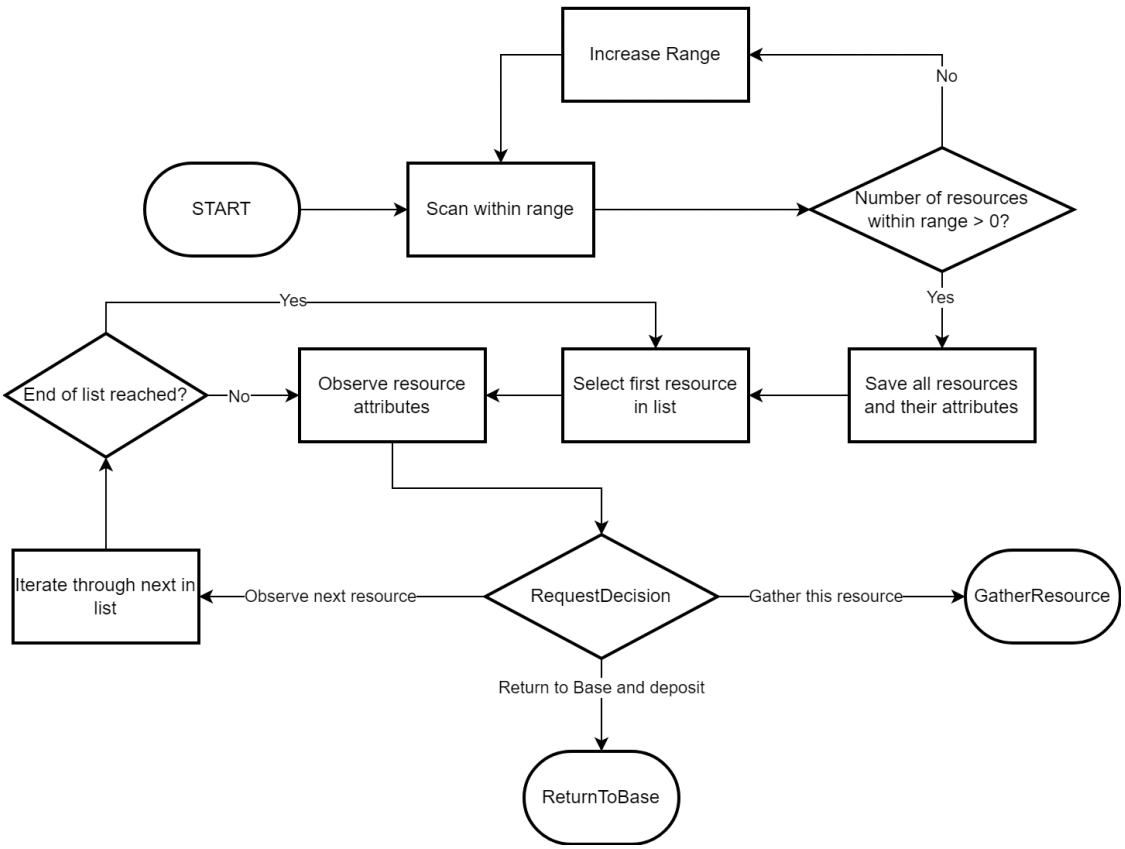

Figure 3.7: *Flowchart of agent's observation and decision process*

3.4.4 Reward and Penalty Setup

For this research, it was decided to use a simple approach for rewards and penalties, with focus on rewarding the agent for score obtained, while also teaching the agent to avoid unnecessary wastes of time.

This was achieved with the following setup:

- Reward for depositing at base

  A reward was given for every resource item deposited at the agent's base. The reward given was based on the amount of the points the resource is worth, resulting in the following reward values for each respective type.

Resource Type	Points worth	Agent Reward
Wood	**5**	**0.25**
Iron	**15**	**0.75**
Gold	**30**	**1.5**

Table 3.4: *Points and Agent Reward per resource type*

- Repeated penalty for walking

  The agent was repeatedly penalised during moments where it was not interacting with any resource or its base, during which it was likely traveling. The weight of this penalty was 0.05 given every second. This penalty was implemented to motivate the agent to minimise unnecessary travel distance.

Before settling on this system, hereafter referred to as Version 3, two other reward/penalty setups were implemented and tested on this agent.

- Version 1

  Rewarding for base deposit, same as Version 3, but without any penalties. This simple system was tried to give the agent a clear goal of obtaining the largest possible amount of points.

- Version 2

  Rewarding for base deposit, but also penalising based on the amount of inventory left empty when deciding to deposit at base. This was attempted to motivate the agent to fill its inventory before depositing at base, minimising number of trips to base.

3.4.5 Agent Training

Training the agent was done on a version of the game without a human-controlled character and their base. The time scale during training was quicker than real-time to allow for faster training, being set to the Unity ML-Agents default of twenty times faster than real-time. Training episodes ended after five minutes to simulate a real game, scaled according to the time scale, and the level reset with newly spawned resources and obstacles to allow for a new episode. The configuration file for the training included a *checkpoint_interval* parameter, allowing models to be extracted without need for the researcher's intervention.

Tensorboard was used to assess agent performance during training, allowing the researcher to view cumulative reward per episode, as well episode length.

Tensorboard also allowed the logging of custom variables. Therefore, the agent's score at the end of every episode was logged, and this was used when deciding which extracted models to carry out further evaluation on.

3.4.6 Model Selection

The agent's training runs were halted once its average score per episode stopped increasing. It was then decided at which points to select the three models for the difficulties by observing the score. The models used for easy and hard were chosen at the points where the scores were the lowest and highest respectively. The medium difficulty model was selected at the midpoint between the lowest and highest score.

Version 1

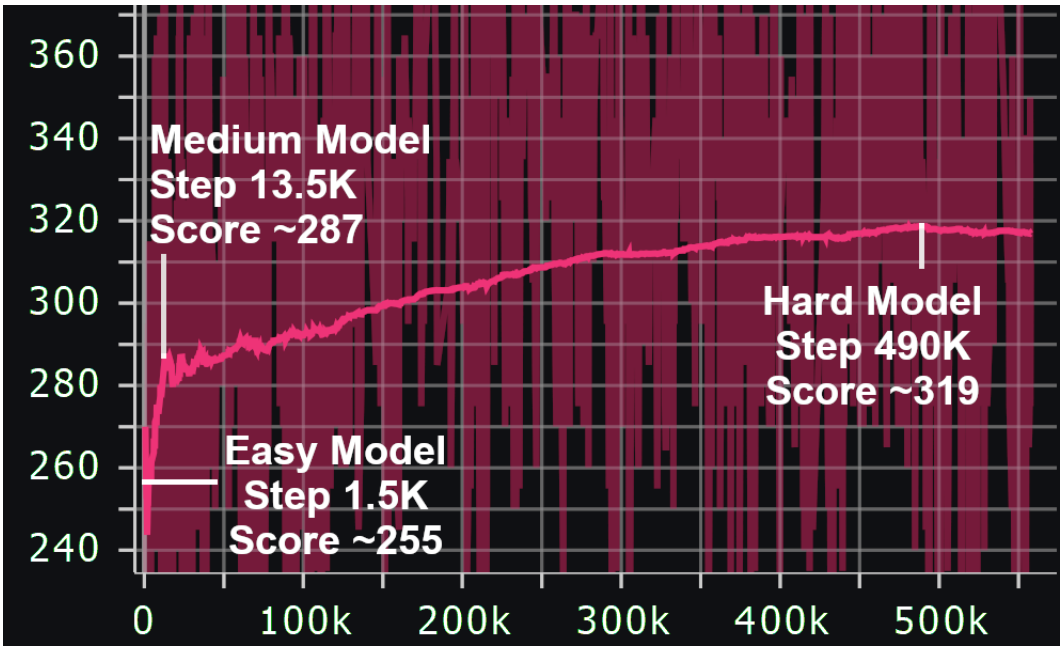

Figure 3.8: *Tensorboard graph of Version 1 agent's average score throughout training, marked with model extraction points*

Training for Version 1 was halted after 145 hours of training, equal to over 557,600 training steps.

Models for the difficulty settings were extracted at the points listed in Table 3.5.

Difficulty Setting	Steps	Time	Average Score
Easy	**1.5K**	**19 minutes**	**255**
Medium	**13.5K**	**9.5 hours**	**287**
Hard	**490K**	**133 hours**	**319**

Table 3.5: *Extraction points for Agent version 1 models*

Version 2

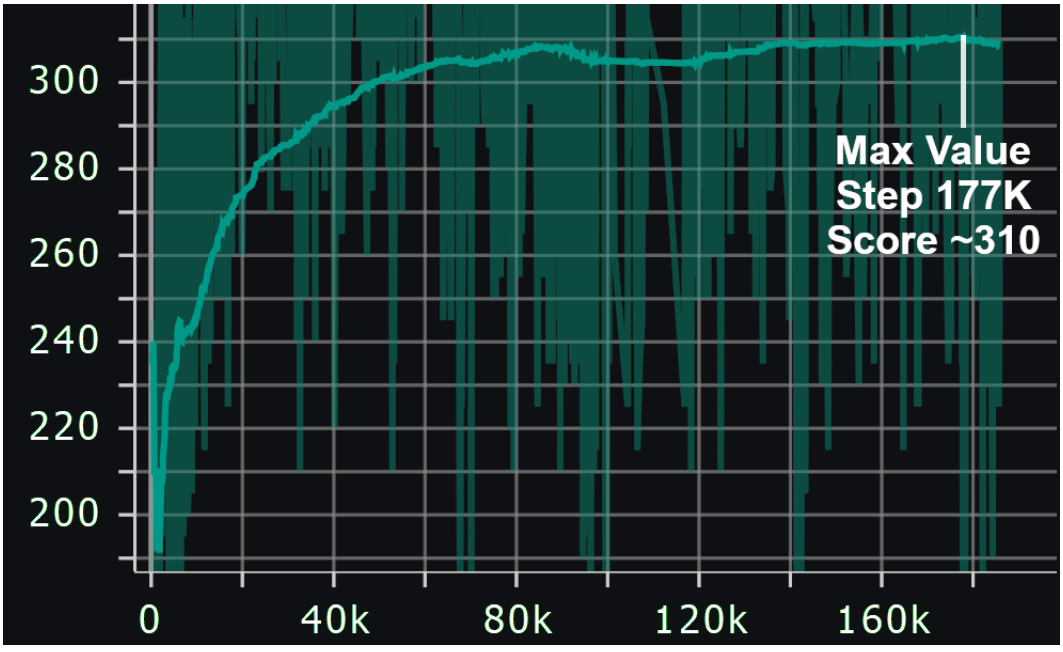

Figure 3.9: *Tensorboard graph of Version 2 agent's average score throughout training*

Version 2 was halted after 40 hours and 185,800 steps of training, during which it achieved a maximum average score of 310. Models were not extracted

for evaluation for this agent, due to poorer performance compared to the previously developed Version 1.

Version 3

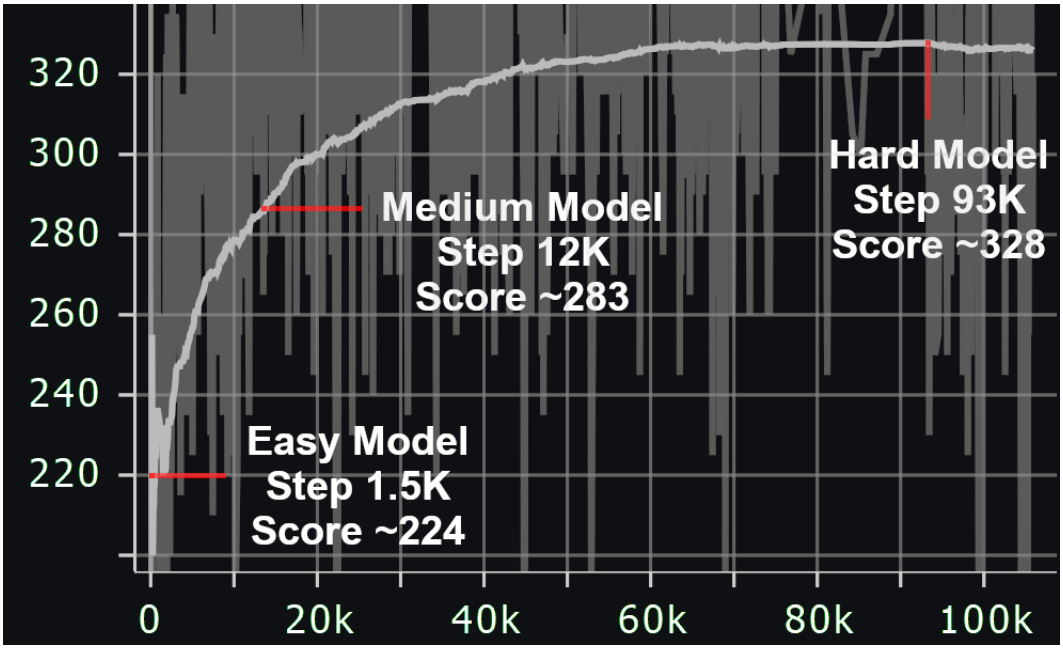

Figure 3.10: *Tensorboard graph of Version 3 agent's average score throughout training, marked with model extraction points*

Training for Version 3 was halted after 16.5 hours and 105,900 steps of training. Models for the difficulty settings were extracted at the points indicated in Table 3.6.

Difficulty Setting	Steps	Time	Average Score
Easy	**1.5K**	**23 minutes**	**224**
Medium	**12K**	**2.7 hours**	**283**
Hard	**93K**	**14 hours**	**328**

Table 3.6: *Extraction points for Agent Version 3 models*

3.4.7 Model Evaluation

To evaluate the performance of models in real-time and examine differences between them, another level was created similar to the training level. The evaluation level also lacked the player character and base, but ran in real-time.

After loading the models wished to be evaluated into the project, the evaluation level was run. On running, an agent is assigned the first model that was loaded into the project, then plays a set amount of five-minute rounds, keeping track of the score on each round. After these rounds are played, the average score of the agent using that particular model is calculated and saved, and the process repeats using the next loaded model, until all the models that require evaluation are used. The results of all rounds and the average score per model were saved in a local file within project.

When evaluating the models for each of the runs, it was decided to have each model play four games before calculating the average, minimising the impact caused by the randomness of resource and obstacle spawning.

Version 1

When running the evaluation level with the Version 1 models, the scores shown in Table 3.7 were achieved for the extracted models.

Model	Round 1	Round 2	Round 3	Round 4	Average Score
Easy	**340**	**460**	**365**	**340**	**376.25**
Medium	**410**	**425**	**450**	**370**	**413.75**
Hard	**415**	**450**	**420**	**490**	**443.75**

Table 3.7: *Score achieved by Agent Version 1 during evaluation*

The difference between models for this agent was promising, but it was decided to develop more versions in an attempt to achieve even better scores. This prompted the creation of Versions 2 and 3 of the agent.

Version 3

The scores shown in Table 3.8 were achieved when running the evaluation level with the extracted models for Version 3 of the agent.

Model	Round 1	Round 2	Round 3	Round 4	Average Score
Easy	**390**	**290**	**400**	**380**	**365**
Medium	**405**	**445**	**385**	**405**	**410**
Hard	**425**	**440**	**470**	**475**	**452.5**

Table 3.8: *Score achieved by Agent Version 3 during evaluation*

Due to the higher average score on the hard model, as well as the overall greater difference between the average scores of the three models compared to Version 1, it was decided to use the models from this agent for the final difficulty settings.

3.5 Data Collection

It was decided to use a mixed-methods approach for this research. Mixed-methods research makes use of both quantitative and qualitative methods with the core assumption that the use of both forms returns more detailed insight than using just one [43]. In this study, quantitative data was extracted from gameplay sessions to analyse agent and player performance, as well as qualitative data from participants giving detailed opinions on their experience playing the game. It was

believed that the combination of these two would provide the clearest picture of how the agent's performance evolved and how it, in turn, affected the final player experience.

3.5.1 Quantitative Metrics

To understand the performance of the agent against a human player, a class was created named *DataLogger* to keep track of all important statistics regarding the game. After a five-minute game concluded, all statistics were saved as files in a custom path set from the main menu.

The files saved per game are the following:

- Player Log
- Enemy Log
- Game Summary
- Performance Summary

Player Log and Enemy Log

The player and enemy logs, as seen in Figure 3.11, show a record of all noteworthy events taken by the player and enemy respectively during the game. Logged events are successful resource interaction, specifying the type of resource, and successful base depositing. Entries in the log are listed in the chronological order in which the event occurred, and even include a precise date and time of occurrence.

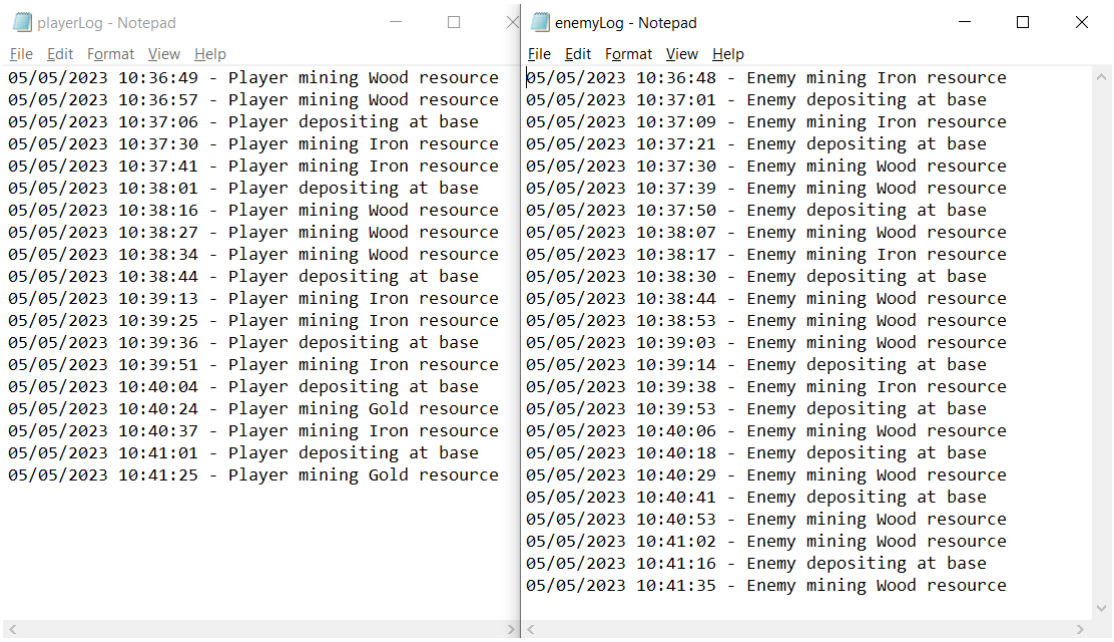

Figure 3.11: *Logs showing both player and enemy actions during the duration of a game*

Game Summary

The game summary, shown in Figure 3.12, stores all important numerical data about the performance of both the enemy and player during the game. This includes keeping track of how many times respectively the player and enemy carried out noteworthy events such as resource interactions and base depositing. It also saves the time in seconds they spent travelling, i.e not interacting with any object, as well as the final scores. Outside of numerical data, the game difficulty is additionally saved in this file.

Performance Summary

The performance summary, seen in Figure 3.13, gives an indication of the computer's performance when running the game by keeping track of relevant metrics during playtime. These metrics are the CPU usage in percentage as well as total

	Player	Enemy
Wood resource interactions	5	11
Iron resource interactions	6	4
Gold resource interactions	2	0
Base deposit interactions	6	9
Time Spent Travelling (seconds)	140	95
Final score	385	380
Difficulty	Easy	

Figure 3.12: *Game summary file showing important game information*

memory used by the game in megabytes. These are saved every second until the end of the game.

Time (seconds)	Cpu Usage (%)	Memory Usage (MB)
1	21.48	77
2	7.62	78
3	14.26	78
4	31.45	79
5	19.53	79
6	9.18	79
7	11.33	80
8	12.7	79
9	10.35	79
10	9.77	79

Figure 3.13: *Performance Summary file showing computer performance metrics in the first ten seconds of a game*

Data Processing

To evaluate difficulty, data from the game summary file was used, calculating average scores and decisions per difficulty for both the agent and the players; therefore, being able to compare the players' performance with that of the agent,

as well as comparing the agent to itself in different difficulties. Sometimes, data from specific games was also used to compare with the opinions of participants, checking both the game summary and log files.

3.5.2 Testing Session and Qualitative Data

The sample of participants was selected from a population of undergraduate IT students, as they could be readily available to attend the testing which took place in one their school's classrooms. From the people contacted, nine male individuals agreed to participate in the study.

Ethical Considerations

To gather data, a testing session was hosted with the volunteering participants. All participants were given a letter of information detailing the aims of the study, what they were expected to do, and how their data would be used. As seen in Appendix D, the letter informed the participants that anonymised images of them may be taken, their speech would be recorded and transcribed, and that their data may be published in anonymous form. It was made clear that all their data would be kept confidential and that they were free to drop from the study at any time, in which case, their data would not be used. They were also allowed to ask questions at any time. All participants signed a consent form agreeing to partake in the study.

Testing Session

Each participant was assigned a computer already loaded with the prototype game. They were given between twenty and thirty minutes to play the prototype, being asked to read the tutorial before starting their first game. Participants were also asked to play through each difficulty setting at least once, but were free to do so in whatever order they pleased. They were also free to replay a difficulty setting within their allocated time if they so wished.

All games played during the testing session generated the previously described files with the quantitative metrics, which were saved to USB drives connected to every computer. Directly after the allocated play-time, the researcher collected these drives from every computer. The researcher later copied all files to his personal computer, sorted them by both difficulty and participant, and created backups.

Focus Group

Qualitative study concerns itself with detailed data based on participant opinions and experiences [44]. As such, this angle was additionally adopted to provide more detailed and accurate results about difficulty perception than with purely quantitative statistics.

To gather this qualitative data, the participants were invited to attend a focus group right after testing to the discuss their experience playing the prototype. A focus group was chosen as it yields direct quotations from participants regarding their experience, similar to an interview [44]. A focus group also allowed par-

ticipants to elaborate on and argue against points brought up by others, resulting in more detailed data. Additionally, it also allowed data to be collected from all participants at once, saving the participants' and researcher's time.

Discussion was guided by questions asked by the researcher regarding things like their player experience and their perception of the game's difficulty, but participants were allowed to speak freely and discuss amongst themselves. The questions prepared by the researcher can be seen in Appendix E. All discussion was recorded using the researcher's mobile phone and later transcribed for use in this study. The full transcription can be read in Appendix F.

Chapter 4: Analysis of Results and Discussion

4.1 Introduction

Following the testing session with the nine participants, the quantitative data of 32 games was successfully gathered. From these games, quantitative analysis was carried out on one game of each difficulty setting for each participant, totalling 27 games. The game selected was the participant's first game played in the selected difficulty setting, unless a technical issue was encountered, in which a later game of that difficulty was used.

This chapter starts out by using the scores for both the player and the agent, extracted from the generated game summary files, and compares them across difficulty settings to get an understanding of whether they are distinct from each other and present an adequate challenge. Following that, the strategies of both the players and the agent are compared with each other and their evolution through difficulty levels is studied. A section is then dedicated to opinions of participants detailing their perceptions of the game's difficulty. Furthermore, performance analytics are analysed to reveal any differences in memory and CPU usage between difficulty settings. Lastly, limitations uncovered during testing are documented, along with possible improvements to the prototype.

4.2 Final Scores

4.2.1 Scores per Difficulty

Easy

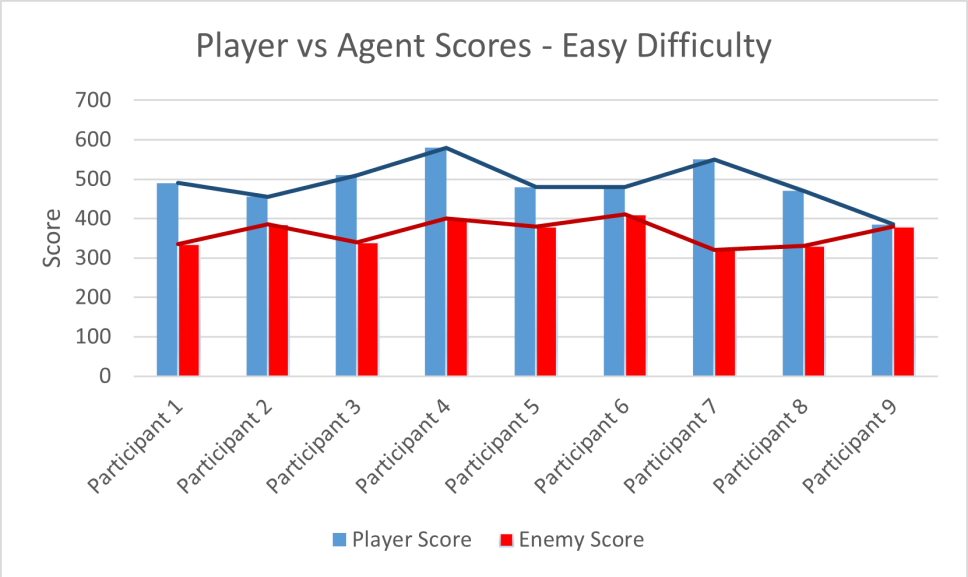

Figure 4.1: *Comparison of Player and Enemy scores on Easy difficulty*

In the Easy difficulty setting, a comparison of the scores achieved by both the players and the agent can be seen in Figure 4.1.

In this setting, the participants scored an average just shy of 489 points, while the agent gained an average score of around 364 points. All games in this setting were won by the participants.

Medium

In the Medium difficulty setting, the participants and the agent achieved the scores per game seen in Figure 4.2.

In this difficulty setting, the participants achieved 540 points on average, while

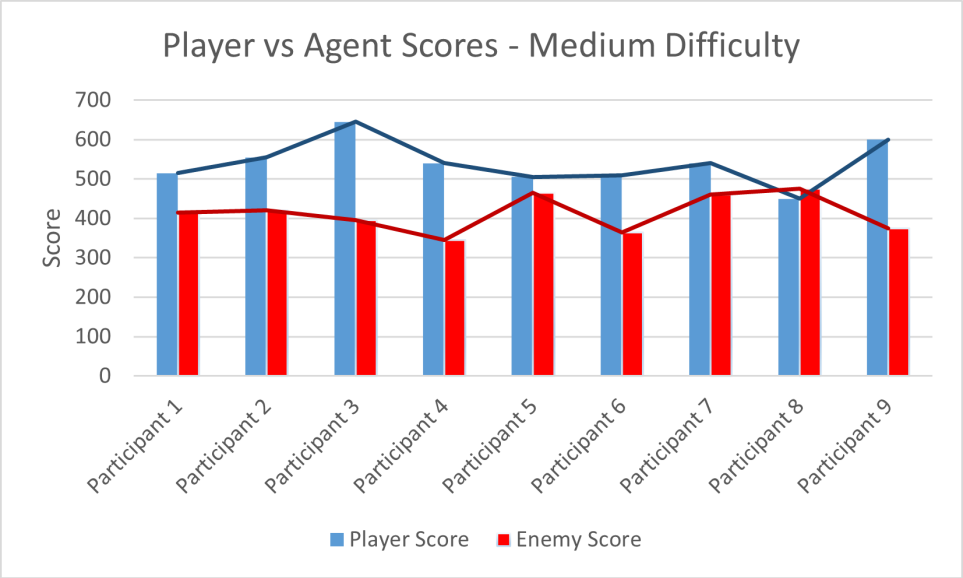

Figure 4.2: *Comparison of Player and Enemy scores on Medium difficulty*

the agent scored an average of 413. Eight of the nine games were won by the participants. The one game lost was by Participant 8, in which the agent won by 25 points.

Hard

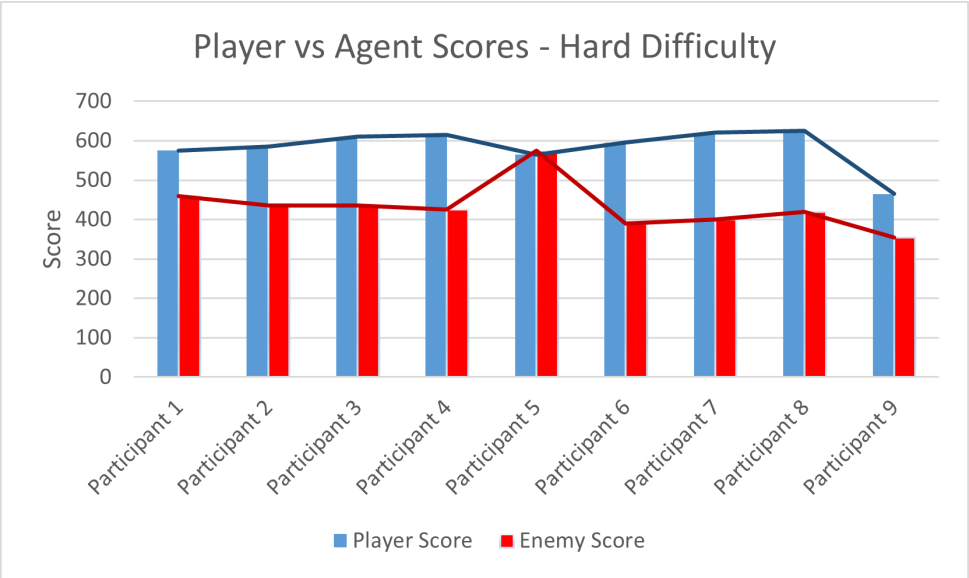

Figure 4.3: *Comparison of Player and Enemy scores on Medium difficulty*

Using the hard difficulty setting, the participants and the agent gained the

scores that can be observed and compared in Figure 4.3.

Playing this difficulty setting, the participants scored, on average, around 584 points, while the agent scored 433. Eight of the nine games were won by the player. The one game won by the agent was against Participant 5, with a difference of 10 points.

4.2.2 Difficulty Analysis

Comparing the average performance, it can be observed that, moving in sequential order through difficulty settings, the agent saw an increase in score, supporting the findings of [8], in which an agent saw improved performance as its intelligence was switched out to a version trained for longer. However, it can be noted that there was a significantly greater increase between the Easy and Medium difficulties, with a difference of 49, in contrast to the difference of 20 between Medium and Hard.

This finding was supported by the opinions of some of the participants. Three of the participants reported that they felt an increase in difficulty between Easy and Medium, but could not as easily distinguish between Medium and Hard.

A few participants also replayed one or two difficulty settings. Some of these stated that even between games of the same difficulty setting, the difficulty did not seem to always remain consistent. This, however, may have been influenced by the element of randomness in the game, such as in the resource spawning, as was suggested by Participant 1.

By comparing player and agent scores, the percentage of difference could be calculated for every game. A positive value indicates a difference in the player's

favour, a negative one in the agent's favour.

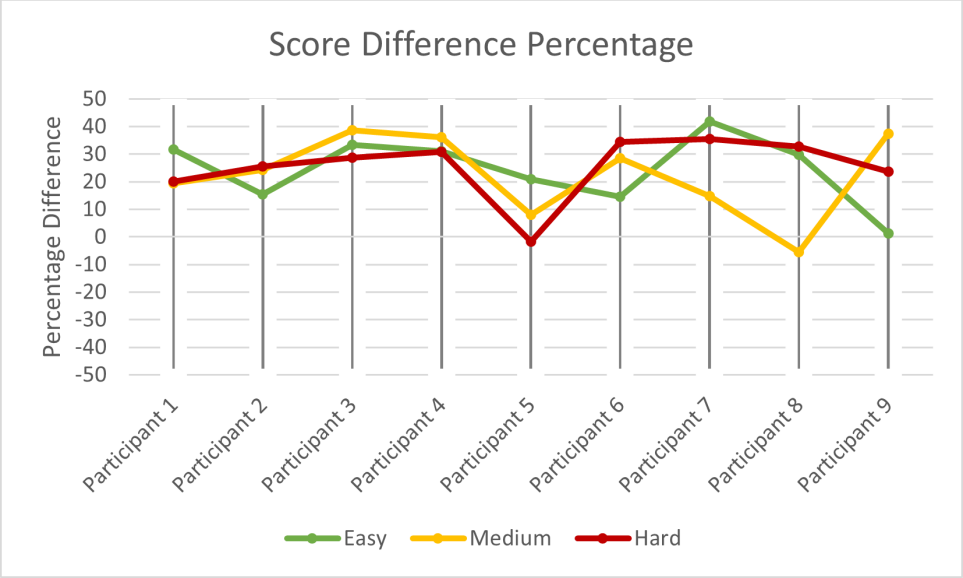

Figure 4.4: *Percentage score difference between player and agent in each game*

Averaging the score difference for each difficulty, it was found to have remained relatively consistent, despite the agent's increasing score.

Difficulty	Average Score Difference
Easy	**24.41%**
Medium	**22.41%**
Hard	**25.54%**

Table 4.1: *Average score difference in percentage throughout difficulties*

The reason behind this was the difference in player scores between difficulties. While participants were allowed to play in whatever order they wished, many preferred to experience the game with sequential increases to difficulty, and therefore decided to play Easy, Medium, and finally, Hard. As such, participants were learning how the game works and finding more optimal strategies while the agent was also incrementally becoming smarter.

It should be noted that score difference was significantly in the player's favour

for all difficulties, instead of being close to 0%, an indication of perfect parity between the player and agent. [45] states that an agent that comes close to achieving parity with the player provides the most adequate challenge. Therefore, this suggests that no difficulty setting served as a true challenge, with some, if not all, difficulty levels benefiting from a boost in difficulty.

4.3 Strategies During Gameplay

4.3.1 Player Strategies

When asked about their play style and preferred strategies, many participants admitted that their strategies changed as they played the game. Most of them reported that, in their initial games, they started out by collecting the closest resources to their base, but eventually started preferring to gather higher value resources, even if they were at a greater distance.

This change can be observed in Figure 4.5, where the popularity of Wood, the most common resource, decreased as difficulty increased. Gold, although rare, was collected in greater amounts in harder difficulties. In the Hard difficulty, it was gathered more than twice the amount it had in Easy, and it surpassed even Wood.

Some participants further elaborated on how they were influenced by factors other than the value of the resources when deciding what to gather. Participant 1, for instance, said that in later games, time was an important factor. For most of the game, he would prioritise searching for and gathering Gold even if it was a considerable distance away. Upon seeing that he had a minute and half left, he

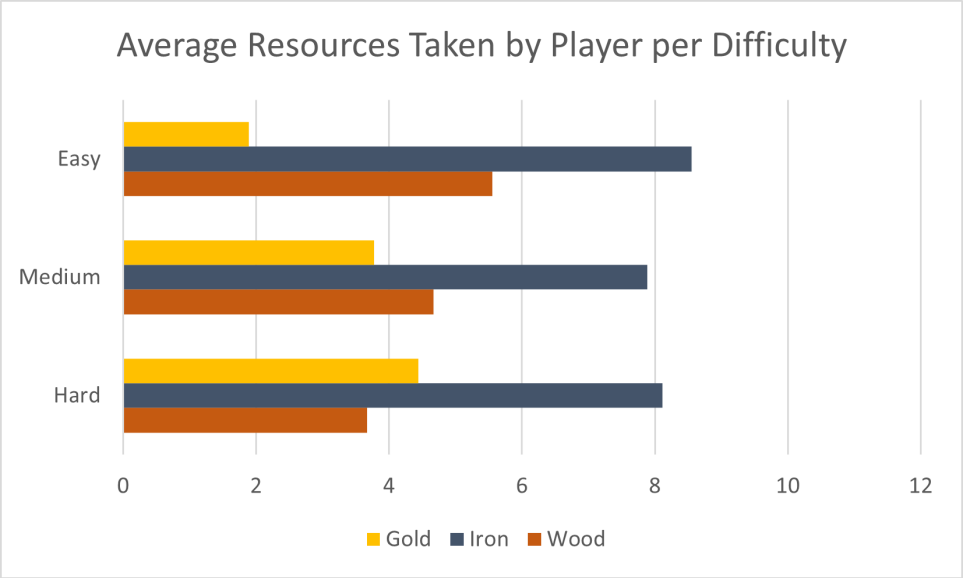

Figure 4.5: *Average amount of resource types gathered by player in each difficulty*

changed his tactic and went for Iron closer to his base. Then, in the last 30 seconds, changed strategies yet again to quickly gather the Wood closest to his base. This behaviour was supported by the logs, such as shown in Figure 4.6. Other participants stated that their choices were influenced by their inventory. They preferred to fill up their inventory to the maximum, even at the expense of choosing a lower-value resource or leaving the resource not fully gathered.

Another interesting strategy was used by Participant 6, who, upon a full inventory, did not immediately return back to his base. Instead, he went around searching for instances of Iron and remembering their locations, so that he could go back to them directly after depositing.

Analysing the number of times players interacted with their base to empty their inventory reveals that it saw no significant difference between levels, remaining at a constant average of 7.44.

Observing total travel time, it was revealed that players spent, on average,

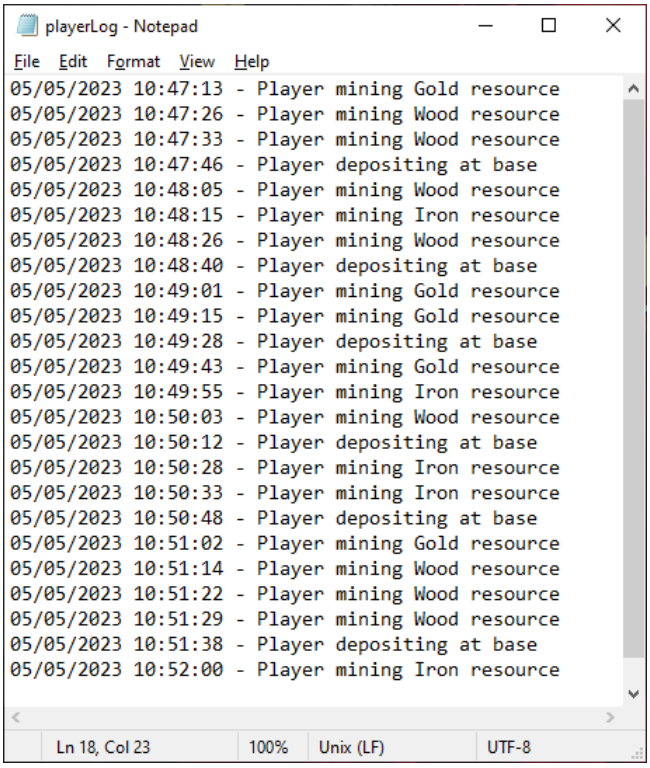

Figure 4.6: *Player log from Participant 1 in the Hard difficulty, gathering Gold commonly in the initial minutes, and focusing on Wood in the last few seconds before the final deposit*

111.44 seconds traveling in both the Easy and Medium settings. In the Hard setting this was lower, at 110.89. However this difference is negligible, with only a difference of around half a second.

From the nine participants, only one admitted to being influenced by the agent's performance; this being Participant 1. If he saw that the agent was catching up in score, he would gather Iron, observing that it gave the most amount of points in the shortest time. Otherwise, he was more lenient in what he gathered, choosing resources that he didn't necessarily view as of the highest value, such as Wood.

4.3.2 Agent Strategies

When players were asked if they could perceive any difference in the agent's strategy between difficulties, only one Participant answered in the positive. This participant felt that in the Medium and Hard difficulties, the agent preferred going more for Iron and Gold. Most of the other participants, however, admitted that they did not continually observe the agent, a few stating that the the agent was not encountered very often due to the way the game was set up. Therefore, changes in behaviour may have went undetected.

Analysing the resources gathered by agent, it can be observed that there were, indeed, changes in its strategy between levels. Similar to the players, the amount of Wood gathered by the agent decreased as the difficulty level increased. In the Medium setting, it was observed that Iron gathering saw its highest level, while in the Hard setting, Gold reached its highest.

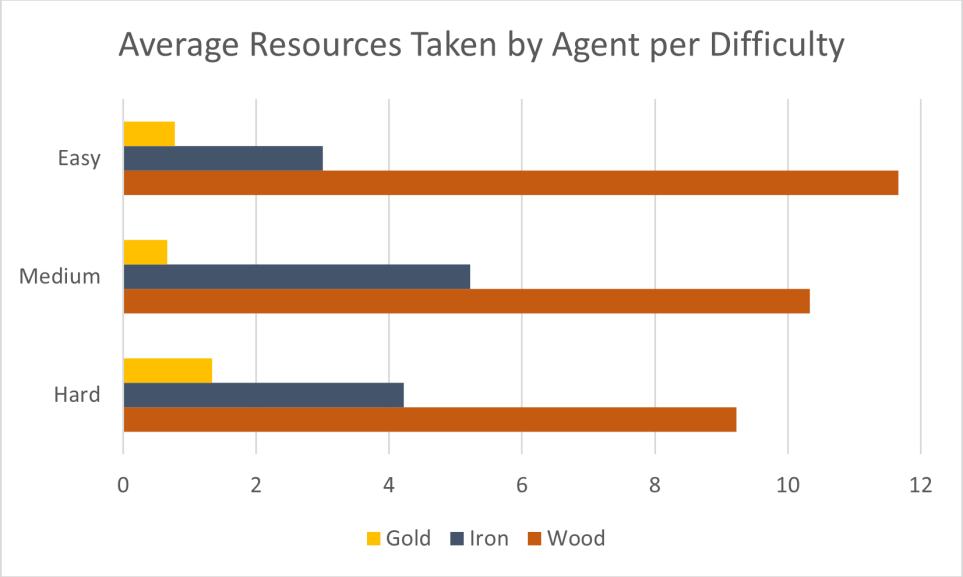

Figure 4.7: *Average amount of resource types gathered by agent in each difficulty*

Despite this, it was noted that the agent's changes were not as drastic, as

can be seen by comparing Figures 4.5 and 4.7. Unlike the players, the order of popularity between resource types remained consistent between difficulties. The opinions of participants generally concur with these findings. Participants 1 and 2, for instance, stated that whenever they observed the agent, it would usually choose Wood over Iron or Gold.

Analysing the number of times the agent returned to its base revealed the following averages per difficulty.

Easy	Medium	Hard
9.22	**8.11**	**8.89**

Table 4.2: *Average agent base deposits per difficulty*

While these figures present distinct differences between difficulties, they do not show a general trend of increase or decrease. They are, therefore, inconclusive to determine strategy changes. It should be noted, however, that all these values are higher than the universal average of the players' 7.44. This indicates that while players may have focused on achieving a full inventory before returning to base, as was admitted by several participants, the agent did not share this tactic, consequently returning to base more frequently. An example of this can be observed in Figure 4.8.

Time spent traveling by the agent was also analysed, and revealed the following average values per difficulty setting.

Easy	Medium	Hard
92.89	**87.56**	**89.89**

Table 4.3: *Average agent travel time per difficulty*

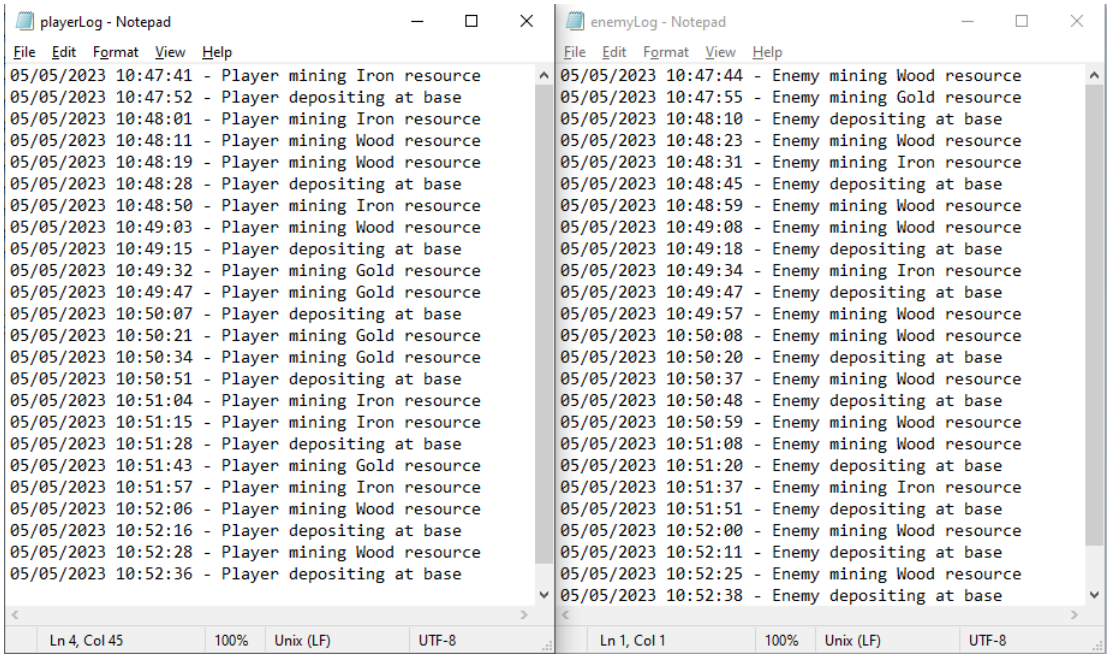

Figure 4.8: *Logs from Participant 7's Hard game. The player usually fills his inventory to a greater degree than the agent, who returns to base more often.*

While differences can be perceived, especially between the Easy and Medium difficulties, a general trend cannot be observed, therefore an evolution of strategy cannot be determined. What can be observed, however, is that these values are generally lower than those of the players'. This suggests that the agent chose to gather resources closer to it, as opposed to the players' strategy of going for more valuable resources at the cost of a greater travel distance.

4.4 Player Perception of Difficulty

During the focus group, participants were asked a few questions regarding their perception of the game's difficulty and left to discuss amongst themselves. When asked if the game had a difficulty setting that served to perfectly challenge them, many participants answered that there was. Participant 8 stated that Medium would be very challenging for new players such as himself. Participants 1 and

5 both had positive things to say about all the difficulty settings. They stated that the levels provided incremental increases in difficulty, which went alongside their own learning process. Participant 3 was the most critical. He felt that all of the difficulty levels were too easy, stating that even the hard difficulty could have been much harder. This supports the previous finding that none of the difficulty levels served to provide a genuine challenge, as was suggested by the score differences.

When asked if there was a setting that was unfairly difficult, the participants unanimously agreed that there was no such difficulty setting. Many participants similarly agreed in the negative when asked if there was a difficulty setting that was, on the inverse, too easy any type of player. The one participant that disagreed was Participant 3 yet again. He felt that in his first time playing the Easy setting, it was incredibly easy. He stated that, at one point, he stopped actively trying to gain points and wandered around the map, yet still managed to win the game by a significant lead. Although this discrepancy in opinion may be due in part to differing skill levels between players, it may also support the previous finding of a lack of consistency within the same difficulty.

4.5 Alternate Approaches to Difficulty

When asked about a possible implementation of a system for automatic difficulty adjustment, participants unanimously agreed that such a system would benefit the game. It was remarked that such a system could benefit both experienced players, by constantly increasing the difficulty for a uniform level of challenge, as well as

new players, in which the system would decrease the difficulty while they would still be getting used to the mechanics. When asked for their ideas about when the difficulty should be modified in the potential system, participants suggested systems based on points. One participant suggested that changes could be based on whether the agent or the player get a considerable amount of points in a short amount of time, a similar approach to the rate of progress implementation by [39]. Another suggested modification based on the difference between the player and agent's scores, increasing the difficulty if it is detected that the agent's score is significantly lower than the players, supporting the study done by [38], where adjustments were carried out based on player and agent comparison. Other participants agreed with this possible implementation.

Participants also suggested other manners in which difficulty could be modified, aside from the modification of agent intelligence. Most of these suggestions revolved around the game's procedurally generated map. Multiple participants suggested that the modification of the resource spawns could be a viable way of altering difficulty, such as by increasing the rarity of the more valuable resources in higher difficulty levels. Another participant also suggested making use of the obstacle spawning system, as he found that the way obstacles spawned could lead to complicated paths that were difficult to traverse.

4.6 System Performance

From the extracted data regarding system performance, the changes in CPU and Memory usage during the game duration were able to be studied per each diffi-

culty.

4.6.1 CPU Usage

Calculating the average CPU usage of every participant per difficulty, it was found that CPU Usage did not see any considerable changes throughout game time.

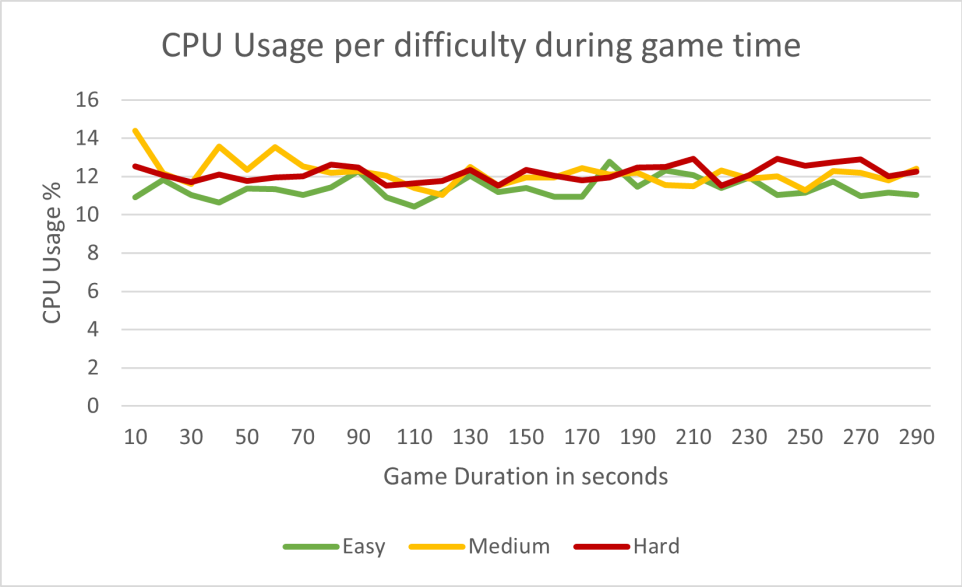

Figure 4.9: *Average CPU Usage during game time*

Additionally, there were no distinct differences in CPU usage between difficulties. Averaging all values throughout game time per difficulty, the following result was achieved.

Easy	Medium	Hard
11.44%	**12.18%**	**12.15%**

Table 4.4: *Average CPU Usage per difficulty*

At most, the difference between these values is 0.74%, which can be considered negligible.

4.6.2 Memory Usage

Analysing average memory usage per difficulty, it was found that the all difficulties had a gradual increase in memory usage throughout game time. This increase, however, proved to be negligible. The difference was, at most, 1.73MB, seen on the Easy difficulty.

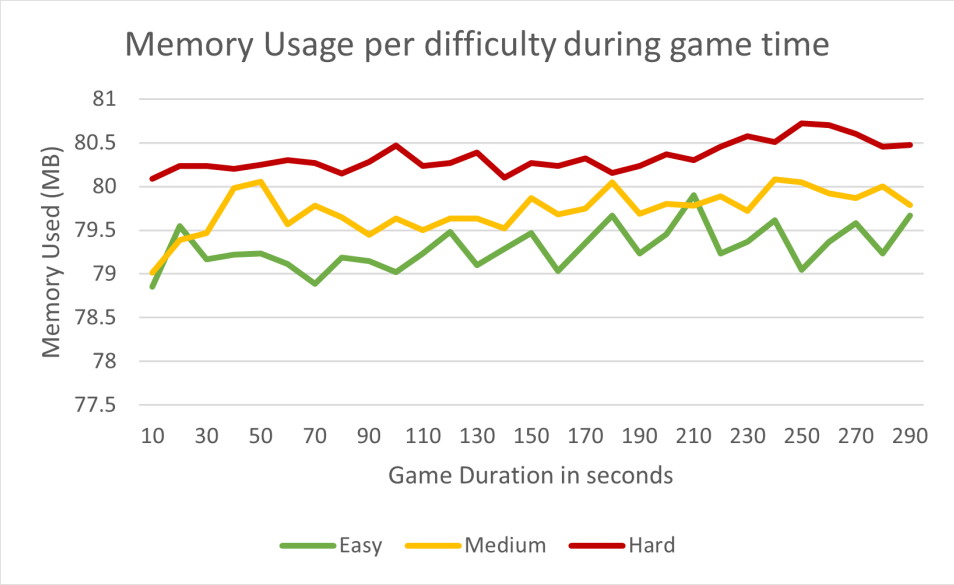

Figure 4.10: *Average Memory Usage during game time*

Calculating the average memory usage throughout game time, the following result was achieved.

Easy	Medium	Hard
79.26 MB	**79.7 MB**	**80.32 MB**

Table 4.5: *Average Memory Usage per difficulty*

The biggest difference was between the Easy and Hard difficulties at 1.06 MB. This difference, however, can also be viewed as negligible.

Overall, these results indicate that there is no significant impact in system performance when using models with different training durations.

4.7 Limitations and Improvements

A number of limitations were revealed during the testing done by the participants; namely, in the behaviour of the agent. It was observed from these findings that these generally occurred whenever the player and agent attempted interaction on the same resource object. In one such game by Participant 4, the agent attempted to interact with a resource the participant was already gathering. This resulted in the agent getting permanently frozen for the rest of the game's duration, hence why this game was not used in any calculations. The results of this specific game can be seen in Appendix C. Another participant, Participant 3, also noticed that by quickly gathering the resource that the agent was heading towards, the agent would still attempt to finish the path towards its original destination, and upon reaching it, would pause for a few moments before moving towards another resource. Participant 3 admitted that, in one game, he attempted exploiting this for his own benefit, disallowing the agent to gather resources.

Many other participants also had suggestions that they believed would make the game more engaging. Several of these suggestions concerned the resources. Participant 9 suggested a "weight" mechanic, reducing the player speed upon gathering heavier resources such as Iron. Most participants agreed with this change, with Participant 8 elaborating on how going for higher-value resources had very few disadvantages, and this would further balance the game. Participant 3 also suggested changes to resource gathering durations, as he felt that the current implementation did not have too distinct a difference between differing-value resources, and so the choice of resource was rather obvious. Another feature was

brought up by Participant 5, who stated that the player should be able to "cancel" his actions; for instance, only gathering some of the inventory items from a resource object, instead of waiting for the resource to deplete or interruption from a full inventory. Other participants concurred with this, stating that they did not enjoy being locked into doing one action when they could quickly move to another to gain an advantage.

One other considerable issue the participants discussed was that they wished to observe the agent's actions, but as was previously found, this could not be accomplished easily. A few ideas were generated to address this issue, some suggesting live statistics, others suggesting a small display showing the agent. Participants were of the opinion that observing the agent's actions may, in turn, influence theirs.

4.8 Discussion Summary

This section analysed the performance of an agent in this procedurally generated strategy game and how it changed as its intelligence was modified throughout difficulties, as well as how players perceived these changes and their impact on the overall difficulty. Answering this study's first research question, it was seen that the final version adopted of the training setup proved effective at teaching an agent how to play on a procedurally generated map, with the final extracted version achieving a better performance than the first. Using automatic periodic extraction during training, a selection of versions of the agent were generated, answering the second research question. Three of these versions were chosen

and assigned to different difficulty settings. These three versions were shown to perform differently and improve over harder difficulty settings, although the rate of improvement was not consistent. Easy to Medium showed a considerable improvement of 49 points on average, but Medium to Hard gained less of a difference at 20 points. Participants commented on this, saying they could easily distinguish between Easy and Medium, but less so between Medium and Hard. It was also found that as players played the game they improved alongside the agent, but usually scored considerably higher, with an average difference of between 22-26% in the players' favour across all difficulties. To address the final research question, many did not seem to be bothered by the high score difference, claiming that they found an enjoyable level of challenge. However, one participant in particular stated that at no point did he feel the game was optimally challenging. Overall, however, this study's hypothesis holds true, with this implementation being a viable way to create difficulty settings in a procedurally generated strategy game.

Chapter 5: Conclusions and Recommendations

5.1 Overview of Research

The aim of this research was to evaluate the effectiveness of using RL and DRL technologies to generate difficulty levels in a strategy game with procedural level generation, achieving this through the modification of an agent's intelligence via the use of different versions of the agent extracted at different points during training. Quantitative analysis proved concrete differences in the agent's performance between levels of difficulty, supporting the findings of [8] when using a similar methodology. Qualitative data also suggested that participants did notice some changes, and many were able to find a decent level of challenge.

From this study's nine participants, all of them won on the Easy setting, eight of them won on Medium, and eight of them won on Hard. Although many saw a considerable jump in difficulty between Easy and Medium, fewer noticed the difference between Medium and Hard. This was reflected in the agent's average score, achieving an increase of 49 moving from Easy to Medium, but increasing with only 20 from Medium to Hard. There were also some complaints about the overall level of difficulty, believing it to less challenging than it should be. This was supported by the finding of the average score difference, being considerably in the player's favour in each difficulty.

It can, therefore, be concluded that the approach explored has the capability of creating distinct difficulty settings in this type of game, but more research

may be needed to create a greater distinction between some of these settings, as well as ensuring that an adequate level of challenge can be found by any type of player.

5.2 Limitations

This study was limited mainly due to time and resources. Due to the resource intensity required to train agents and the researcher's sub-optimal hardware, agents could not be trained quickly. Therefore, only a small number of versions could be developed, trained, and tested within the limited time frame. As such, it may be possible that the final version of the agent used in this study is not the most ideal. A different version of the agent's observations technique may be able to give a clearer picture of its environment, and an alternate rewards and penalty setup may prove fruitful in getting the most out of the agent.

Owing to the limited time available and limited personnel who could test the agent, there were also bugs and unwanted behaviours that were revealed in the agent's logic during testing and were brought up by the participants.

Participants also suggested manners in which the prototype could be optimised for testing. This included reworking elements such as the resource gathering times to make them more balanced, as well as new features such as a possible weight mechanic that would give the game an additional layer of strategy. Further care may also be taken to allow players to observe the agent more easily so that they can provide more detailed feedback about its actions, as well as take them into consideration when making their own choices.

This study was also limited in its sample of participants. Since participants were all IT students, it may be believed that all had a considerable level of experience playing games. Therefore, it could not be fully determined how these difficulty levels can be perceived between participants of vastly different gaming experience. Participants were also all male and of similar ages, so it could not be determined how these demographic factors may affect difficulty perception.

5.3 Recommendations for Future Work

It is recommended that, for future research, the agent should undergo more thorough development and rigorous testing with the aim of developing a more ideal version that can create a greater distinction between difficulty settings, while simultaneously minimising bugs and logic errors.

It is also recommended that future studies use a larger and more demographically diverse sample of participants, using participants of different experience in playing video games, as well as different ages and genders. This would provide a more detailed picture of how difficulty settings can suit any type of player.

Additionally, the researcher's main recommendation would be to attempt a study on how a similar system can be used alongside DDA in order to shape a more enjoyable player experience, adjusting the agent automatically during the course of a single game, as opposed to a premeditated change done manually by the player. It may also be beneficial to explore how agent intelligence can be modified in conjunction with a game's procedurally generated elements, such as obstacles in map traversal and rarity of valuable items, for more engaging ways

with which to create challenging, yet enjoyable, difficulty settings.

List of References

[1] Mihaly Csikszentmihalyi. *Flow: The Psychology of Optimal Experience*. Harper-Perennial, 1991.

[2] K Fathoni, R Y Hakkun, and H A T Nurhadi. Finite state machines for building believable non-playable character in the game of khalid ibn al-walid. *Journal of Physics: Conference Series*, 1577(1):012018, Jul 2020.

[3] Richard S Sutton and Andrew Barto. *Reinforcement learning : an introduction*. The Mit Press, 1998.

[4] Georgios N. Yannakakis and Julian Togelius. *Artificial Intelligence and Games*. Springer, 2018.

[5] Steve Rabin. *Introduction to game development*. Course Technology Cengage Learning, 2010.

[6] Le Lyu, Yang Shen, and Sicheng Zhang. The advance of reinforcement learning and deep reinforcement learning. *2022 IEEE International Conference on Electrical Engineering, Big Data and Algorithms (EEBDA)*, Feb 2022.

[7] OpenAI, Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique Pondé de Oliveira Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon

Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. Dota 2 with large scale deep reinforcement learning. 2019.

[8] Frank G. Glavin and Michael G. Madden. Skilled experience catalogue: A skill-balancing mechanism for non-player characters using reinforcement learning. In *2018 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 1–8, 2018.

[9] Josef Wiemeyer, Lennart Nacke, Christiane Moser, and Florian 'Floyd' Mueller. *Player Experience*, pages 243–271. Springer International Publishing, Cham, 2016.

[10] J. Mccarthy and Peter Wright. *Technology as Experience*, volume 11. The MIT Press, Sep 2004.

[11] Judeth Oden Choi, Jodi Forlizzi, Michael Christel, Rachel Moeller, MacKenzie Bates, and Jessica Hammer. Playtesting with a purpose. *Proceedings of the 2016 Annual Symposium on Computer-Human Interaction in Play*, Oct 2016.

[12] Robin Hunicke. The case for dynamic difficulty adjustment in games. In *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology - ACE '05*. ACM Press, 2005.

[13] Joy James Prabhu Arulraj. Adaptive agent generation using machine learning for dynamic difficulty adjustment. *2010 International Conference on Computer and Communication Technology (ICCCT)*, Sep 2010.

[14] Anubhav Anand and Ajay Kumar. The rise of artificial intelligence in video games. *International Research Journal of Modernization in Engineering Technology and Science*, 4(7):2768–2771, Jul 2022.

[15] Qiyue Yin, Jun Yang, Kaiqi Huang, Meijing Zhao, Wancheng Ni, Bin Liang, Yan Huang, Shu Wu, and Liang Wang. Ai in human-computer gaming: Techniques, challenges and opportunities. *arXiv:2111.07631 [cs]*, 14(8), Aug 2022.

[16] John E Hopcroft, Rajeev Motwani, and Jeffrey D Ullman. *Introduction to automata theory, languages, and computation*. Pearson, 2014.

[17] Mat Buckland. *AI game programming by example*. Wordware ; Lancaster, 2004.

[18] Risto Miikkulainen, Bobby D. Bryant, Ryan Cornelius, Igor V. Karpov, Kenneth O. Stanley, and Chern Han Yong. Computational intelligence in games. In Gary Y. Yen and David B. Fogel, editors, *Computational Intelligence: Principles and Practice*. IEEE Computational Intelligence Society, Piscataway, NJ, 2006.

[19] Martijn van Otterlo and Marco Wiering. *Reinforcement Learning and Markov Decision Processes*, pages 3–42. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[20] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4, 1996.

[21] Amitha Mathew, P. Amudha, and S. Sivakumari. Deep learning techniques: An overview. In Aboul Ella Hassanien, Roheet Bhatnagar, and Ashraf Darwish, editors, *Advanced Machine Learning Technologies and Applications*, pages 599–608, Singapore, 2021. Springer Singapore.

[22] Kevin Gurney. *An introduction to neural networks*. Ucl Press, Cop, London, 1997.

[23] Lei Chen. *Deep Learning and Practice with MindSpore*. Springer Singapore, 2021.

[24] Nicole Rusk. Deep learning. *Nature America*, 13(1):35, Jan 2016.

[25] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, Nov 2017.

[26] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning, 2013.

[27] Vlad Firoiu, William F. Whitney, and Joshua B. Tenenbaum. Beating the world's best at super smash bros. with deep reinforcement learning. *CoRR*, abs/1702.06230, 2017.

[28] Inseok Oh, Seungeun Rho, Sangbin Moon, Seongho Son, Hyoil Lee, and Jinyun Chung. Creating pro-level ai for a real-time fighting game using deep reinforcement learning. *IEEE Transactions on Games*, 14(2):212–220, 2022.

[29] Anthony DiGiovanni and Ethan C. Zell. Survey of self-play in reinforcement learning, 2021.

[30] John K Haas. A history of the unity game engine. 2014.

[31] Maxwell Foxman. United we stand: Platforms, tools and innovation with the unity game engine. *Social Media + Society*, 5(4):205630511988017, Oct 2019.

[32] Arthur Juliani, Vincent-Pierre Berges, Ervin Teng, Andrew Cohen, Jonathan Harper, Chris Elion, Chris Goy, Yuan Gao, Hunter Henry, Marwan Mattar, and Danny Lange. Unity: A general platform for intelligent agents. *arXiv preprint arXiv:1809.02627*, 2020.

[33] Adi Aiman Bin Ramlan, Azliza Mohd Ali, Nurzeatul Hamimah Abdul Hamid, and Rozianawaty Osman. The implementation of reinforcement learning algorithm for ai bot in fighting video game. In *2021 4th International Symposium on Agents, Multi-Agent Systems and Robotics (ISAMSR)*, pages 96–100, 2021.

[34] Nirmal Baby and Bhargavi Goswami. Implementing artificial intelligence agent within connect 4 using unity3d and machine learning concepts. *International Journal of Recent Technology and Engineering (IJRTE)*, 7(6S3):193–200, Apr 2019.

[35] Muhammad Iftekher Chowdhury and Michael Katchabaw. Bringing auto dynamic difficulty to commercial games: A reusable design pattern based approach. In *Proceedings of CGAMES'2013 USA*, pages 103–110, 2013.

[36] Mohammad Zohaib. Dynamic difficulty adjustment (dda) in computer games: A review. *Advances in Human-Computer Interaction*, 2018:1–12, Nov 2018.

[37] Su Xue, Meng Wu, John Kolen, Navid Aghdaie, and Kazi A. Zaman. Dynamic difficulty adjustment for maximized engagement in digital games. In *Proceedings of the 26th International Conference on World Wide Web Companion*, WWW '17 Companion, page 465–471, Republic and Canton of Geneva, CHE, 2017. International World Wide Web Conferences Steering Committee.

[38] Johan Hagelback and Stefan J. Johansson. Measuring player experience on runtime dynamic difficulty scaling in an rts game. In *2009 IEEE Symposium on Computational Intelligence and Games*, pages 46–52, 2009.

[39] Mirna Paula Silva, Victor do Nascimento Silva, and Luiz Chaimowicz. Dynamic difficulty adjustment on moba games. *Entertainment Computing*, 18:103–123, 2017.

[40] Robin Hunicke and Vernell Chapman. Ai for dynamic difficulty adjustment in games. *Challenges in game artificial intelligence AAAI workshop*, 2, Jan 2004.

[41] M.I. Chowdhury and M. Katchabaw. Improving software quality through design patterns: A case study of adaptive games and auto dynamic difficulty. *13th International Conference on Intelligent Games and Simulation, GAME-ON 2012*, pages 41–47, Jan 2012.

[42] Gustavo Andrade, Geber Ramalho, Hugo Santana, and Vincent Corruble. Extending Reinforcement Learning to Provide Dynamic Game Balancing. In

IJCAI 2005 Workshop on Reasoning, Representation, and Learning in Computer Games, pages 7–12, Edinburgh, United Kingdom, July 2005.

[43] J.W. Creswell and J.D. Creswell. *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches*. SAGE Publications, 2017.

[44] Michael Quinn Patton. *Qualitative Research*, volume 3, pages 1633–1636. John Wiley & Sons, Ltd, 2005.

[45] Gustavo Andrade, Geber Ramalho, Alex Sandro Gomes, and Vincent Corruble. Dynamic game balancing: An evaluation of user satisfaction. In *Proceedings of the Second AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, AIIDE'06, page 3–8. AAAI Press, 2006.

Chapter A: Prototype Link

Link to Unity Project:

`https://github.com/GeorgGrech/GeorgGrech_ThesisProject.git`

Chapter B: Game Summary Tables

Participant	Player	Agent
Participant 1	**490**	**335**
Participant 2	**455**	**385**
Participant 3	**510**	**340**
Participant 4	**580**	**400**
Participant 5	**480**	**380**
Participant 6	**480**	**410**
Participant 7	**550**	**320**
Participant 8	**470**	**330**
Participant 9	**385**	**380**

Table B.1: *Player and Agent scores in Easy difficulty, used to generate Figure 4.1*

Participant	Player	Agent
Participant 1	**515**	**415**
Participant 2	**555**	**420**
Participant 3	**645**	**395**
Participant 4	**540**	**345**
Participant 5	**505**	**465**
Participant 6	**510**	**365**
Participant 7	**540**	**460**
Participant 8	**450**	**475**
Participant 9	**600**	**375**

Table B.2: *Player and Agent scores in Medium difficulty, used to generate Figure 4.2*

Participant	Player	Agent
Participant 1	**575**	**460**
Participant 2	**585**	**435**
Participant 3	**610**	**435**
Participant 4	**615**	**425**
Participant 5	**565**	**575**
Participant 6	**595**	**390**
Participant 7	**620**	**400**
Participant 8	**625**	**420**
Participant 9	**465**	**355**

Table B.3: *Player and Agent scores in Hard difficulty, used to generate Figure 4.3*

	Hard	Medium	Easy
Wood	**3.666667**	**4.666667**	**5.555556**
Iron	**8.111111**	**7.888889**	**8.555556**
Gold	**4.444444**	**3.777778**	**1.888889**

Table B.4: *Average amount of resource types gathered by player in each difficulty, used to generate Figure 4.5*

	Hard	Medium	Easy
Wood	**9.222222**	**10.33333**	**11.66667**
Iron	**4.222222**	**5.222222**	**3**
Gold	**1.333333**	**0.666667**	**0.777778**

Table B.5: *Average amount of resource types gathered by agent in each difficulty, used to generate Figure 4.7*

Chapter C: Results from Erroneous Game

	Player	Enemy
Wood resource interactions	1	4
Iron resource interactions	9	2
Gold resource interactions	5	0
Base deposit interactions	8	2
Time Spent Travelling (seconds)	132	20
Final score	575	130
Difficulty	Hard	

Figure C.1: *Game summary file from erroneous game, showing unusually low values*

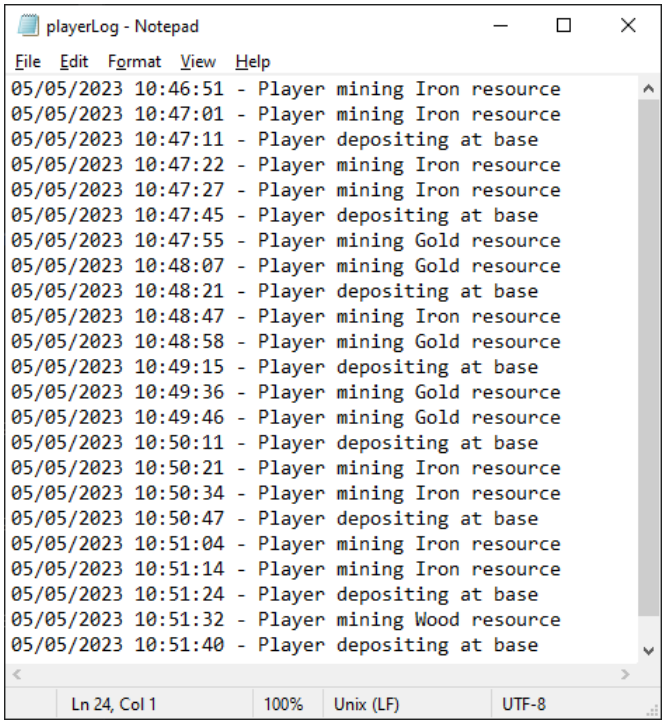

Figure C.2: *Player Log file showing Participant 4's actions during the erroneous game, spanning the entire five minutes*

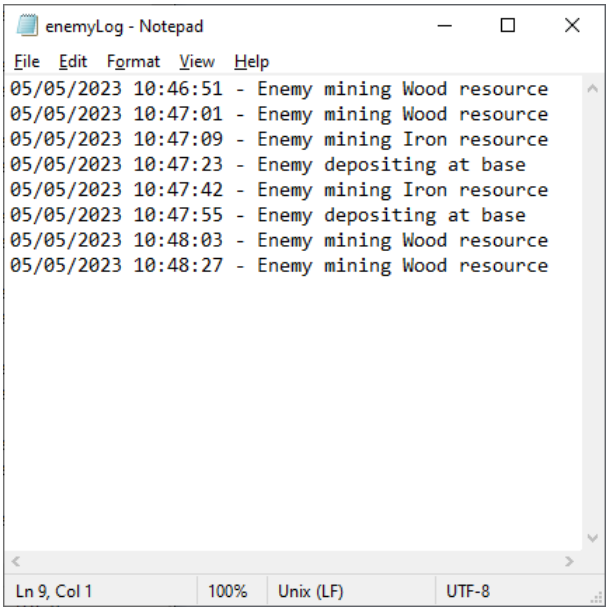

Figure C.3: *Enemy Log file showing agent's actions during the erroneous game, halting less than two minutes in*

Chapter D: Participant Letter of Information

Participant Letter of Information

Title of Research: Creating difficulty levels with reinforcement learning in a strategy game

Name of Researcher: Georg Grech

Researcher Contact Email: georg.grech.d57175@mcast.edu.mt

You are being invited to take part in a research study. Before you decide to participate, it is important for you to understand why the research is being done and what it will involve. Please take time to read the following information carefully and discuss it with others if you wish. Ask us if there is anything that is not clear or if you would like more information.

This research is being undertaken on the use of reinforcement learning in casual strategy games to create difficulty levels by adjusting intelligence of in-game opponents. You have been chosen because of your experience with video games and interest in the field of game development.

It is up to you to decide whether to take part. If you decide to take part, you will be given this information sheet to keep and be asked to sign a corresponding consent form. You will be asked to join a session in which you will play 15-30 minutes of a prototype game, during which anonymised images of you may be taken. Directly afterwards, you will participate in a focus group with other participants where you will be asked questions regarding the prototype.

Your answers will be recorded and transcribed for use in this research.

There are no disadvantages or risks foreseen in taking part in the study, and by taking part you will be contributing to the development of a set of recommendations for the creation of difficulty levels in video games with the use of reinforcement learning technology.

If you wish to complain or have any concerns about any aspect of the way in which you have been approached or treated during this study, please contact the researcher via email. All information which is collected about you during the research will be kept strictly confidential so that only the researcher carrying out the research will have access to such information and will not be shared with any other individuals. Participants should note that data/images collected from this project may be retained and published in an anonymized form. By agreeing to participate in this project, you are consenting to the retention and publication of data.

The results of this research will be written up into a dissertation for the researcher's final project of his Bachelor of Science (Honours) in Multimedia Software Development degree, provided by the Malta College of Arts, Science and Technology (MCAST) Institute of Information and Communication Technology.

If you would like more information about the research before you decide you may contact the researcher via email at the address supplied above.

Thank you for your interest in this research.

Chapter E: Focus Group Preparatory Questions

1. Overall, how was your general experience playing this game?
2. Did you perceive there to be a considerable difference between the difficulty levels?
3. Was there a difficulty level that served to perfectly challenge you?
4. Was there a difficulty level that you felt was unfair?
5. On the other hand, was there a difficulty that was too easy for anyone?
6. Did your play style change as you played the game? What where your preferred strategies throughout?
7. Did you notice if the opponent's preferred strategy changed according to the difficulty?
8. Did you notice any patterns in the opponent's behaviour that remained consistent?
9. Do you think this game would benefit from a system that automatically adjusts the difficulty?
10. In that system, when should difficulty be changed?
11. Do you have any suggestions on how the difficulty levels can be improved?

12. Any final observations or comments you would like to make regarding the difficulty settings or the game itself?

Chapter F: Focus Group Transcription

Researcher: OK. Before we begin, I would like to ask everyone if they understand that they're being recorded and their answers will be transcribed for use in this research.

Multiple Speakers: Yes.

Researcher: OK, so we can continue. So the way this focus group is going to work is I'm going to ask you a few questions about your experience playing this game. And you can answer them one by one, but you can also continue and say something about something someone else said and have a discussion; maybe you want to elaborate on a point someone else mentioned. So feel free to speak your mind whenever. So my first question is overall, how was your general experience playing this game? Did you have a good player experience? Did you enjoy this? You can start one by one.

Participant 7: It was very enjoyable and very simple. I did not particularly hate the game and I did not particularly enjoy the game a lot.

Participant 5: It was quite enjoyable. Quite fun, entertaining.

Participant 6: It's very visually immersing.

Participant 4: It was a pretty polished prototype, given that it is a prototype, and it was actually enjoyable.

Participant 1: Yeah, I agree. It was a pretty polished prototype. And the game mechanics were simple. Yet fun.

Participant 8: Yeah, I think the concept was fun and it was pretty challenging at times.

Participant 2: I think it was pretty interesting, trying to find our strategies. It was pretty fun.

Participant 3: I agree with the others. The prototype was really polished. Good job and that. It was fun, I thought, to try and beat the AI using strategies. I think you did a pretty good job.

Researcher: All right. Thank you. We can move on to the next question. Also feel free to discuss amongst yourselves. Did you perceive there to be a considerable difference between the difficulty levels?

Participant 9: For me, I don't think so. I didn't pay attention to what score the AI was achieving, but I was only focusing on my score, so I didn't pay any attention to … if it was... (in Maltese) Can you repeat the question?

Researcher: If there was a considerable difference between the levels.

Participant 9: For me no, like all levels were...

Participant 7: In my case. From easy to medium, there was a bit of a jump, it was harder. But then from medium, I think medium was harder than hard, for me.

Researcher: So, you thought that medium wasn't very different than hard or it was even more difficult.

Participant 7: Yeah

Participant 5: I remember actually the score range for easy it was 200, the enemy wise for easy was 200. Medium was around 400 and for hard was 500,

so from my experience was a good jump in increase.

Participant 6: For me, the first time round because I've tested the hard twice, they were quite the same, but it could be because I've adapted to the game, I started to learn. But the second time playing the hard difficulty, it was very hard for me. I actually came very close. I've won all games. I don't know if it's the programming, but that's how I've seen it. You've seen the score as well, it was 5 difference.

Participant 4: For me, there was a considerable difference between the difficulties. At times in the higher difficulty, the AI came pretty close to my score. However, in one of the runs, the AI agent broke and I managed to get a pretty significant lead. In the second run in the higher difficulty it was able to catch up to me at one point, but I still managed to win by a difference of 50, 60 points.

Participant 1: Yeah, from my end when I jumped into the easy game it was quite close and it was quite challenging at first to get used to how long it takes to get resource and to put them back into base. So it was quite close, but then going from medium to hard, I noticed myself thinking a lot more. But the scores were similar. So, I don't know. I think it was around 400 for each of the difficulties, but with each difficulty I found myself thinking harder and harder about strategy and stuff like that. So, I do think yes, there was a considerable jump in the in difficulties.

Participant 8: From my experience, first of all I started with medium before I went to easy and it was a bit challenging. In fact, I lost the first round. Then

I decided, I won when I tried again on medium, and then I went to hard and I didn't see any big jump. In fact, I think the bot got less points than he did in medium. And it felt almost evenly in terms of…

Researcher: So, there wasn't a considerable difference between medium and hard?

Participant 8: No, no. And in terms of easy, it was, it felt easier. But I don't think it was noticeable like we wouldn't have noticed that I didn't consciously like change the difficulty myself.

Participant 2: So, for my first game I played on easy and until I actually got the hang of understanding the time frames for each resource, um, the first game was actually pretty close with me and the AI and after the medium went to hard, basically it was not that as difficult as the first time that I played it. And so, I thought I'd go back and play the easy again, and I still for some reason found the easy a bit harder than the other two. But yeah, that's, that's my experience.

Participant 3: So from my end I played easy twice and medium and hard twice because obviously the results. On my first play through easy I find, personally I found it very easy. At one point I was just exploring the map more than gathering resources and I still won by a good lead. Then I played medium and I felt a shift in difficulty the most, I think, but I still won. And I played hard and it was almost the same. Maybe the bot managed to get another hundred points than the medium. And then I played hard again and it was much harder than the than the first hard I played. It was the hardest. The second hard

that I played was the most challenging. At one point, the bot even got some more points, and I managed to catch up.

Researcher: So before moving on, I'm noticing that some of you said that even on the same difficulty there wasn't that much consistency.

Participant 3: No.

Researcher: Sometimes there was a variance.

Participant 3: The second hard I've played was much harder than the first hard I played.

Participant 1: Alright, it could be down to the randomly spawning items and stuff like however.

Researcher: OK so, do you think there was a difficulty overall that served to perfectly challenge you?

Participant 3: I think it could have been harder. The hard, especially the hard, I think could have been much harder than it was.

Researcher: Right.

Participant 9: (in Maltese) Is this a new question?

Researcher: This is a new question, but you don't have to answer each question.

Participant 8: From my end ...can you repeat the question?

Researcher: Was there a difficulty level that served to perfectly challenge you?

Participant 8: Yeah, I think medium is good for new players because it was very challenging and I think the element of randomness in the game helps a lot

with keeping you challenged at times.

Researcher: All right.

Participant 5: I agree. Let's just say for starters, easy is good. Getting used to the mechanics. Medium, if you if it would be the next step. If you want to challenge yourself would be hard, at least for me, the scores were really close.

Participant 1: And I agree with that. From my experience, I experienced the same thing exactly. The drops in difficulty were incremental, which was perfect.

Participant 2: From my end the spawning was very lucky, I think, on the medium and the hard. So that's why the gap between them wasn't that significant. And, in fact I think if the spawning maybe rate was less on the hard it would have actually directly impacted the result.

Researcher: OK so you didn't feel there was overall there was a difficulty that was maybe unfair, completely unfair.

Multiple: No.

Researcher: Alright. So you all think: No, there wasn't an unfair one. What about was there a difficulty that was, maybe, too easy for anyone playing?

Multiple at once: No.

Participant 3: First easy, ... that was, at one point I was just going around the map.

Participant 6: What I think, since it was the first time playing easy, it was quite because I just gathered wood, the closest that was to the spawn point, to the base, and it was hard in the easy ones because I'm not getting the best resources because I didn't know that there was better resources. I think that's

why the easy was quite easy, ah, easy not so easy.

Researcher: So on the point that you mentioned the next question is did your play style change as you played the game? Did you change strategies?

Multiple: Yeah, yes, yes.

Participant 9: For me, I started the easy level, I started gathering material close to the base and after I played the medium level I start wandering searching for gold ores only until the timer was at 3 minutes 30 seconds. After that I start gathering only iron ore close to the base and until the last maybe 30 seconds I just started gathering only wood which was the closest material to the base.

Participant 8: Yeah, for my end, I've also noticed a big change. For example, I started off only gathering wood and then I realized that obviously the other resources get more increments in points. So obviously I decided to always prefer going for the higher increments.

Participant 1: I started off always trying to gather the closest materials, especially because wood and iron take the same amount of space after one, like one dig but wood obviously gives you 20 for the four slots and iron gives you 15, so I was basing that that knowledge and comparing with the AI. If the AI was, you know close to me, I tried to get iron because it was quicker to dig and quicker to deposit. But if I had time and I had quite a lead I would even go for wood. And obviously if I struck gold then, you know, lucky me.

Participant 4: In the beginning I got used to the to the mechanics of the game, but gradually as playing the game I changed that. I changed the strategy, I started roaming for higher income ores such as gold and iron and what

I found to be one of the best strategies is trying to fill up the inventory with as much ore as one can possibly gather, and even if that means leaving an ore unfinished and then returning to the base, putting it, emptying the inventory, and then gathering some more ore.

Participant 2: From my perspective as well, similar to how you managed to do it and the last basically like few slots that were empty, I just filled them up with wood along the way, take them back and basically leave the unfinished so I can like do it. I guess in my mind was like quicker to just have it unfinished and then come back and take it again.

Participant 6: For me it was quite different. I mean, I went to get the winner as iron and then instead of gathering the wood, I went scouting more instead of going back. Therefore, I know where each iron is and then I will, I will get more time going directly to the place where the iron is.

Researcher: OK. So did you notice if the opponents preferred strategy changed according to difficulty?

Participant 1: Couldn't tell.

Participant 3: I wasn't really paying attention to what he was doing. I just saw his points going up so I would know where I stood.

Participant 8: I noticed that it was particularly hard to see what it was doing because you don't see him much around the map.

Participant 5: Yeah you could see it from the score basically that it was preferring, for example, going for gold and iron rather than wood.

Participant 1: And from the few times that I saw the AI, he'd usually go

for wood more than ores.

Participant 2: Same here.

Participant 4: I did pay attention to the increment and scores of the AI. However, it preferred going more for wood and iron that were closer to his base.

Researcher: So did these strategies change according to difficulty or did they remain kind of mostly the same you feel?

Participant 4: They changed, I feel like they changed with difficulty, especially in the medium and hard difficulty where he preferred going more after iron and gold.

Researcher: Alright, now that you've played this game do you think it would benefit from a system that would maybe adjust the difficulty for you automatically?

Participant 1: Definitely. Yes.

Participant 3: Could you repeat the question?

Researcher: Do you think that that this game would benefit from a system that would adjust the difficulty for you instead of you selecting it?

Multiple: Yes.

Participant 8: I agree with that. Especially, like keeping the game fun for the player, constantly upping the difficulty makes it so that you always have a challenge basically.

Participant 4: Or on the other spectrum, if someone is new to the game of the mechanics or they're introduced to them, the game would progressively get easier, or make it easier for the player to maintain the challenge equal to his

skill.

Researcher: So when should the difficulty change according to you? When should it observe that it should up the difficulty?

Participant 4: I believe that it should up the difficulty if it notices an increment of points based on a specific amount of time.

Participant 2: Specifically for the AI as well. If the AI is constantly being much lower than the player, I think it should give the player a bit more challenge. And exciting.

Researcher: So mostly on the difference between the agent and the player.

Multiple: Yes.

Researcher: Do you have any suggestions on how the difficulty system on these different levels could be, maybe improved? Maybe by changing the enemy or even the game itself? How could we have a better difficulty system?

Participant 2: The ore, I think the way that basically like spawns, I think it also might be fruitful to like adjust that based on difficulty as well, like the frequency of the ores themselves.

Participant 8: Yeah, maybe like higher value ores could spawn more rarely in higher difficulty games.

Participant 4: Yes.

Participant 6: Maybe on the top right there should be a map also where you have explored. Basically, if you explore the right side of the of the base, that will show what's on the right side so we don't have to remember that there was ores on the right side.

Researcher: That's a more general game mechanic you're saying, ok.

Participant 3: The terrain affects a lot of the difficulty of the game, like how certain rocks are placed, because you have to go around and you have to plan your route. I've been finding myself, for example, wandering a lot further from the base. Just try and find the gold and stuff.

Participant 1: Yeah, but those who implement the percentage of like higher value ores spawning like closely or close to the base and with each incrementing difficulty that percentage gets lower and lower.

Participant 5: I'm not sure if it's related, but I would implement in a way where both the enemy and you and you will see for example, like you mined wood for example 10. Gold 20. You have like a tag. That way you can see OK, the enemy is going more for gold so I should step up my game.

Researcher: So you have kind of live statistics on what your enemy is doing.

Participant 5: Yeah, basically, yes.

Researcher: OK. All right. So before we conclude. I would just like to ask you if you have any final observations or comments that you'd like to make that weren't related to any questions about the game or the difficulty settings, any general observations?

Participant 9: Maybe about the difficulty, you can do for example if me, the player, gathered iron the player speed will be reduced because the iron is much more heavier than the wood so, there player should change his strategy on what he would gather.

Participant 4: Yeah. So there would be a weight mechanic system.

Multiple: Yeah.

Participant 8: Because I feel like once the player discovers that iron and stuff gives you more points, it's always the obvious solution to go for, so you need to add some sort of disadvantage to balance it out.

Participant 3: For example, even the mining speed, if. I'm not mistaken, the mining speed between wood and iron is really non-significant. So if there is an iron node and a wood node, which iron usually spawns right close to your base, I'm just gonna go eat that iron up and take it back to base.

Participant 8: I think it would also be a cool idea if there was like a split screen so you could see exactly what the enemy is doing so you could see exactly what they're doing. I think it would be very fun.

Participant 1: A small hud more than a split screen.

Participant 8: Yeah, yeah. Anything that would like to show you what the enemy is doing, basically.

Participant 2: I would also like to shed some light on the tutorial as well, which is something that we didn't talk about that much, but from the tutorial itself, we don't exactly get like the perception of how long it would take to mine those particular ores and I think if we had the time to like sort of a playground test with those assets, I think we would have had like a bit of a jump start when we actually started playing the game. I think that would have also significantly like improved....

Participant 6: But the easy difficulty, I mean that's what it was. Basically it was a substitute for that.

Multiple: Yes.

Participant 3: Yeah, I agree. Personally, I had more fun jumping into easy not knowing any of the values and trying stuff for myself than having the games spoonfeeding me the values.

Multiple: True.

Participant 6: But the tutorial was still needed because I skipped the tutorial and jumped straight to the easy one and I was confused.

Participant 3: No, the tutorial does explain the basic objective of the game. That you have to press E, that there's nodes that you have to take back.

Participant 1: Exactly, and then, it's up to you to like, strategize also.

Participant 3: Yes, I think that's nice, but having the games spoon feeding you values of "Listen would this take 5 seconds to mine and could give you..." for me personally...

Participant 4: It's much easier finding a strategy that way and it would kind of remove the enjoyment from the game.

Participant 3: Yeah, I think learning that for yourself, it gives you tiny bit of learning curve.

Participant 4: Even before starting the game, you can make up a strategy to win against the AI. Yeah, you can find one of the most optimized way of gathering points.

Participant 5: Two small features that I think to include is one: when I would show basically the way when you're the base, doesn't go gradually; when you deposit the item, doesn't go one by one, it actually goes, after you finish,

all at once. And another feature is actually I would cancel the deposit.

Multiple: Yeah.

Participant 5: For example, I see that you know he's catching up on points. I would cancel, go immediately then go back.

Participant 6: Yes, yes, I agree with that because sometimes you don't really want everything you just want to gold. The timer is getting down, it's last 10 seconds and you don't want to continue mining the gold, for instance, which takes long, so we just cancel it in the first ore you get.

Researcher: So you want a feature that you can just interrupt your actions.

Multiple: Yeah

Participant 3: I think the most the most reason for it, the biggest one is when there's like 10 seconds, 15 seconds and you're just mining.

Participant 1: The strategy changes.

Participant 3: And for example, the points there and you know that if you cancel it right now, you might get that tiny bit of a lead.

Participant 2: Definitely.

Researcher: So, any more observations or is that it?

Participant 6: That's it from my end.

Participant 4: That would be it.

Researcher: OK, so we can conclude here.

Participant 3: I don't know if, section for bugs.

Researcher: If you have any final things you want to mention about bugs you can mention.

Participant 3: I did find a small bug that if you find the AI and you see what, for example, there's a gold node. And because I think he's got pre-planned paths where he goes and if I take his gold node for like, even a split second before, which I did, he kind of stares at you.

Multiple: (Laughing)

Participant 1: Like, really? Are you stealing my gold? Why would you take my gold?

Participant 3: And I tried like to delve into it on my 4th playthrough and I was like, let's see what can I do. And I was just basically going around seeing what node he was going to and just steal it from him.

Multiple: (Laughing)

Participant 3: And he'd just stare at you.

Participant 1: So that's a new strategy. (laughing)

Researcher: So you tried to purposefully trip up the agent.

Participant 3: Yes, I was. I took the PVE element of it seriously and I was trying to...

Participant 2: It's a strategy.

Researcher: That is a strategy, yeah.

Participant 3: But I think that if his decision time was maybe a bit better, for example if he sees you going for his node and he just switch it, I think that would be...

Participant 2: Maybe he should try and steal your node.

Multiple: (Laughing)

Participant 3: Yeah.

Participant 4: Regarding that. It did happen. In my case, he tried to mine the same node as I was mining at the same time.

Participant 1: Yeah

Participant 4: And he just got stuck as again he found a middle ground. He just left next to a point where it was in between our bases. And he just stood there, even if he tried to mine the same node as I was mining or not me stealing it from him, but him trying to steal it from me.

Participant 3: I could also move him. For example if he's walking and just push him around.

Participant 1: Just mean to the AI.

Participant 3: Basically yes.

Participant 5: I think it would have like a little bit of functionality. You coded in a way that the that the player looks at the mouse. That's like, that's not your fault, I understand. But in a way I would add it is he would mine if he's looking at it. You know, there's like this a stick figure to show where he's looking at. I would do it in a way if he's looking away, he doesn't mine it. I know it's harder for the AI to understand that but for player-wise I would do it that way.

Participant 3: I agree.

Participant 6: You can cancel it anytime.

Participant 5: Like adding more mouse functionality, yeah.

Researcher: Ok yeah.

Participant 1: Or it's more, it's faster or something like that if you look at the ore.

Participant 3: Even if you had like a, I don't know if you played like those old strategy games where you can like get, grab your mouse to like see further. I don't mean that you can see all the map, but if you could, while you're walking like you have your camera just move it a tiny bit. Even if it's black, if it's got dark corners and you can only see, for example, I don't know, 20 blocks.

Participant 4: So, you wouldn't be at the center of the camera at all times. You could move it a bit and you would be like at the edge of the screen.

Participant 3: Yeah, I think that would have been a good functionality.

Participant 6: Because I think the AI already knows where the resources are, right?

Participant 3: Yes.

Participant 6: Or it doesn't work with radius?

Researcher: It scans the area around it and makes a decision.

Participant 4: OK.

Participant 6: I see.

Participant 3: So it doesn't know exactly where all the nodes are?

Researcher: No, it doesn't know beforehand where everything is.

Participant 6: So, the area of scanning is the same as the area of what we're looking at?

Researcher: It's similar, but if it finds nothing in that range, it increases its range.

Multiple: OK, OK, alright.

Researcher: So, any more observations?

Participant 4: No

Researcher: OK, so we can conclude here. I thank you all.

Multiple: Thank you.