Evaluating Player Appreciation towards Dynamic Difficulty Adjustment managed through Artificial Intelligence

Adrian Mercieca

Supervisor: Ivan Briffa

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 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 Ivan Briffa.

05/06/23 .... A. Mercieca ....

Date Signature

Copyright Statement

05/06/23
....
Date

[illegible]
....
Signature

Acknowledgements

I would like to express gratitude to my partner for her unwavering support and love throughout this long and difficult journey. I am truly grateful for her constant encouragement and belief in me.

To my family and friends, thank you for your support and understanding. Your presence and encouragement have been invaluable.

I am grateful to my supervisor, Ivan Briffa, for his guidance. His mentorship has been instrumental in shaping the quality of this research.

I would also like to thank the participants who generously shared their time and insights, contributing to the success of this study.

Abstract

Dynamic Difficulty Adjustment (DDA) is a system that enables real-time difficulty balancing, which is used to elevate the user experience by adjusting the game to match the player's level of skill. Artificial Intelligence (AI) is proving its potency in assisting with every day tasks. AI can be applied to a multitude of tasks such as automating trivial duties to handling high risk real-time systems. In this research a Deep Q-Network (DQN) model is developed and trained to adjust game difficulty depending on the current contrast of lives remaining between two teams at battle with each other. The game on which the system was developed is an FPS shooter named 'Ravenfield'. 20 participants were asked to play both an original and an experimental version of the game, which included the DDA system. Subsequently, they were asked to fill in a questionnaire detailing their experience about each match played. Finally they took part in a short interview which enabled discussion on the relevance and usefulness of DDA. Many found the concept intriguing and applicable to enhance different experiences. The results showed an increase in enjoyment, even though a slight decrease in engagement was registered. More information, including participant opinions, were gathered and discussed within the study. The results of this research, together with other literature could pave the way for game developers to effectively implement a DDA system which benefits and entices the whole gaming community, with veterans and new comers alike.

Keywords: Dynamic Difficulty Adjustment, Artificial Intelligence, Deep Q-Network, Neural Networks.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Network
DQN	Deep Q-Network
DDA	Dynamic Difficulty Adjustment
FSM	Finite State Machine
MCTS	Monte Carlo Tree Search
NN	Neural Network
NPC	Non Player Character
QPD	Quick Progressive Difficulty
RDHP	Relative Difference of players Health Points

Chapter 1: Introduction

Since their inception, video games have rapidly increased in popularity and have become a common type of household and portable entertainment. The difficulty in developing games that are both entertaining and enjoyable to a variety of players is a recurring issue for developers in the game development industry. The use of Dynamic Difficulty Adjustment (DDA) in games is one possible remedy for skill related accessibility issues. This entails automatically modifying the difficulty level of the game in response to the player's performance.

1.1 Research Background

Recently, excellent Language Models have taken the globe by storm and have enabled Artificial Intelligence (AI) to further establish itself as a potent technology. With AI stabilising itself as such a potent technology, this research aims to explore whether real-time player performance statistics could be provided to an AI model that dynamically modifies the game's difficulty to suit the player's capabilities. DDA has been found to boost player satisfaction and engagement, while also making games more accessible for players. Research is required to further understand the spectrum of effects caused by utilising DDA.

1.2 Research Objectives

The purpose of this study is to investigate the effects of utilising DDA. This study specifically looks at an AI model that is implemented through reinforcement learning. An assessment is made on how DDA affects player engagement, enjoyment, and accessibility.

1.2.1 Research Questions

This study aims to answer these research questions:

- Does Dynamic Difficulty Adjustment affect player performance?
- Do players notice when adjustments occur?
- Does adjustment affect the player's enjoyment, frustration, or perception of the game?

1.3 Research Framework

Research Philosophy

'Interpretivism' is selected as the research philosophy since the purpose is to understand the subjective experience and perspective of the players. Qualitative research is used to uncover the complex and nuanced meanings that people attach to their experiences and actions.

Research Approach

An inductive approach is taken because this study's goal is to uncover themes and patterns from the data collected. Since this topic is still relatively unknown, this exploratory approach is ideal to further the knowledge related to DDA, by arriving at a hypothesis through the data itself.

Research Strategy

Data is gathered using an experiment. This experiment compares two versions of the same game, in which one utilises the developed DDA system. The results of these two scenarios are compiled and compared to produce important insights on the particular implementation of DDA used.

Choices

Mixed-Methods are used to gather the necessary data. Participants are asked to answer several questions, some of which are closed-ended (quantitative) and some of which are open-ended (qualitative) and promote further discussion.

Time-Horizon

Data was gathered at a single point in time, from participants possessing varying levels of skill in playing digital games.

Techniques and Procedures

Stratified sampling is used to select participants based on age, as this can directly relate to the participants skill in digital games, where middle-aged people are likely to have significantly less skill in the area than teenagers. Through self-assessment, 11 participants identified as novices and the remaining 9 participant agreed with being casual to expert players.

1.4 Personal Motivation

A desire to dig deeper into the process behind delivering a satisfying experience through gaming was a deep personal motivation of mine. As can be seen within this research, consumers in the market are hard to please. This is not because of excessive consumer demands, and for the most part, neither is it due to developer shortcomings. Consumers in this market are extremely subjective, as one player's demands will usually contradict another player's preferences and therefore a product that satisfies every consumer is highly unlikely. Through research in Dynamic Difficulty Adjustment, developers might have the data and resources available to develop a more personalised experience, such that players of all skill types may enjoy the game at similar levels of enjoyment, engagement and frustration. Frustration is a part of the gaming experience and as seen in this study, it can be a factor which promotes engagement. Therefore, small amounts of sporadic frustration are desirable, whilst large amounts or a common sense of frustration is a deterrent and something Dynamic Difficulty Adjustment can help to tackle.

1.5 Scope of the Study

The study aims to add to the expanding body of knowledge about the use of AI in video games and to gather relevant information about how to set up a DDA system that is both effective and beneficial to a wide variety of end users. The results provide information for game developers that are looking to introduce and appease a wider range of people through their products.

Chapter 2: Literature Review

2.1 Overview

Artificial Intelligence is a vast subject and incorporates many aspects of modern-day technology. Machine Learning is a system that processes as though it is using human-like thinking to arrive at a conclusion. In this manner, it can take decisions and learn without human intervention [1].

2.2 The Origins of AI: A Historical Outlook

2.2.1 John McCarthy and Alan Turing

According to Smith et al. [2], John McCarthy is known for devising the name "Artificial Intelligence". McCarthy states that "The study is to proceed on the basis of the conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it. An attempt will be made to find how to make machines use language, form abstractions and concepts, solve kinds of problems now reserved for humans, and improve themselves." [3]. Although McCarthy invented the term "Artificial Intelligence", the idea was first conceived by Alan Turing which discussed the idea in his paper "Computing Machinery and Intelligence" [4]. Both McCarthy and Turing are known to be pioneers of AI, however Turing has left a lasting impact with his invention of the Turing Test which to this day is used to evaluate the quality of AI models [4].

2.2.2 Alan Turing's Philosophy

Hodges's [4] study of Turing's research identifies a key difference between Turing's different papers years apart. Hodges [4] states that, in the paper released by Turing in 1936, he mentions the possibility of machines having different configurations which he relates to the different states of mind of a human calculator. While this was a powerful idea, he was even bolder in his paper released in 1946, where he mentions the idea of a machine showing intelligence. What Alan Turing is most known for, however, is his paper released in 1950. Hodges [4] explains that in this paper, Turing's foremost argument is that machines would in fact be able to do human actions which require initiative, imagination, judgement, and cause surprise.

2.2.3 The Mathematical Objection

Hodges [4] goes on to explore objections to the field, such as "The Mathematical Objection". He makes note of a statement made by Turing in his paper released in 1946, where he mentions that it is possible to make a machine display intelligence at the risk of it making occasional serious mistakes. The mention of "mistakes" within the paper brought confusion and objection to the prospect of machine learning. He explains that humans, being discrete state machines, can only employ an algorithm. Gödel's theorem states that no algorithm can coincide with truth seeking in every case [4]. Therefore, every algorithm, employed by human or machine is bound to fail occasionally. Hodges's [4] resolution to the Mathematical Objection puts humans and machines on a levelled playing field, something which Turing himself appealed for when calling for the concept of "fair play for machines".

2.3 Different Categories of AI Development

2.3.1 Learning Techniques

Artificial Intelligence and Machine Learning can be trained either by making use of supervised learning, unsupervised learning or through hybrid techniques [1].

2.3.2 Supervised Learning

Skinner and Walmsley [1] and Mohammed et al. [5] explain that supervised learning occurs when using labelled data, meaning that a supervisor is aware of what the expected output should be. Supervised Learning can occur through the use of machine learning techniques such as Decision Trees, Artificial Neural Networks and Support Vector Machines, amongst others [6]. The algorithm of AI that most accurately mimics how the human brain works is the Neural Network (NN). The structure of NNs is based on the knowledge we have on how the human brain functions. Zahedi [7] details the structure of NNs as having Layers, Nodes, Connections and Weights. There must be an input layer and an output layer, but a NN can contain additional hidden layers which augment the input data. He goes on to explain that each layer has a number of Nodes, which depending on the input they receive, generate an output. Nodes within various layers are connected via Connections. These connections have weights that determine the input value given to the next node. Skinner and Walmsley [1] go on to divulge that Neural Networks act as a series of switches that activate only if the received input is close to the desired value. This is all based on how the human brain uses neurons to function.

2.3.3 Unsupervised Learning

On the other hand, Mohammed et al. [5] explain that in unsupervised learning, the data being used to train a model is not labelled, hence there is no target output. In Unsupervised Learning, similar data is grouped together forming what is commonly referred to as a Cluster [8].

2.3.4 Hybrid Learning

With the existence and use of Supervised and Unsupervised Learning, there also exist Hybrid Techniques such as Reinforcement Learning [1]. Reinforcement Learning is a technique which is used in different aspects of life, such as child development, or pet training. This entails rewarding or punishing the subject depending on whether the desired action was performed. In Machine Learning, a model undergoes the same treatment, by receiving feedback based on its output. Schaeffer [9] explains that the agent learns through trial and error to optimize actions, with the aim of maximising reward values. Skinner and Walmsley [1] explain that this continuous feedback allows it to learn to derive the best possible output.

Deep Q-Network

Reinforcement Learning has been proven to be effective when utilising Q-Learning [10]. Q-Learning is based on the use of a Q-Table which stores the maximum expected reward for each action at each state. By comparing the relevant Q-Values in the Q-Table, the action with the highest reward is given preference.

More recently, Deep Q-Networks have been introduced. This Reinforcement Learning method is gaining popularity and is especially being used to create smarter NPCs within digital games. In a Deep Q-Network, a number of Q-Values are approximated using a Convolutional Neural Network [11]. Compared with the Q-Table lookup in standard Q-Learning, this algorithm can be considerably faster when handling larger state spaces, therefore, it is advantageous in scalability [12]. In Deep Q-Networks a Q-Function is used to represent action-state values. The Convolutional Neural Network which approximates the target value is periodically updated by the Q-Function to stabilize the learning process [11].

2.3.5 Comparison of Learning Techniques

Kotsiantis [6] explains that Supervised Learning attempts to teach a machine what is known, by providing labelled data as its input. Researchers however are making use of unsupervised learning to detect that which is unknown, by identifying patterns and recognising useful classes of items. Hybrid Techniques will learn both what is known and what is unknown due to trying many possible actions in an attempt to maximise rewards.

2.4 AI and Video Games

2.4.1 The Introduction of AI to Board Games

Yannakakis and Togelius [13] explain how games have long been used as a research field for Artificial Intelligence. Even before the conception of AI, early pioneers of computer sciences developed programs specifically aimed at playing games. They intended to examine whether computers could complete activities that appeared to require “intelligence”. Alan Turing (re)invented the Minimax algorithm and used it to play Chess, which along with Checkers, Yannakakis and Togelius [13] describe as the games on which such research was mostly based. A. S. Douglas created the first piece of software to successfully master a game in 1952 while working on a digital version of the board game Tic-Tac-Toe.

2.4.2 The Introduction of AI to Video Games

The first video games were created without any form of integrated AI. These games were simple and with the main intention being to engage real people in competition [1]. The author states that early Atari games such as Pong, introduced AI’s use in video games. Schaeffer [9] explains that unlike simple board games, video games often have real-time limitations that block players from thinking deeply about each move. Unpredictability obstructs players from fully planning future events, and concealed information prevents players from fully knowing what other game players, whether AI or Human, are doing.

2.4.3 AI Achievements

A major landmark in the history of AI playing video games happened in 1997 when IBM's Deep Blue exhibited extraordinary Chess playing proficiency. The Deep Blue model used a Minimax algorithm and harnessed the power of a supercomputer to defeat the reigning grandmaster of Chess, Garry Kasparov. Smith et al. [2] state that AI hasn't experienced rapid progress since its early studies and advances. The authors wondered whether this was due to its complex nature or a lack of understanding of the field through basic research. AI in video games had also suffered from slow progression, which Skinner and Walmsley [1] attribute to a lack of motivation from game developers. However, more recent research shows that in 2017, a new milestone for the area was achieved by OpenAI, who had managed to develop an AI that consistently beat human players at a MOBA game called DotA 2 which pits teams of five players against each other with the objective of advancing within the map and destroying the opposing team's base [1]. The AI prevailed when put head-to-head against a professional player and soon after, it was successful in beating a team of 5 semi-pro players which were ranked in the 99.95th percentile of skill [1].

2.4.4 Human vs AI

Today, some of the best DotA 2 players have been beaten by machines equipped with deep learning algorithms [1]. Skinner and Walmsley [1] go on to claim that the possibilities of Artificial Intelligence and Deep Learning are endless and that scholars suggest autonomous training for Neural Nets is one of the expected outcomes of future research. This will result in a larger number of intelligent machines which according to [1], might result in professional players being trained by the very machines that will have surpassed them.

2.4.5 Progress of Research on AI in Video Games

In Serafim et al. [14], released in the early 2000's, the authors highlight a recurring and still relevant problem for AI development in the Games Industry: AI is not a selling point in the games market [1]. Game developers have been mainly focused on graphical improvements, tying to make games which are aesthetically beautiful and realistic. This has been necessary in a highly competitive market with clients consistently demanding aesthetic enhancements. Serafim et al. [14] also note that graphical improvements were slowing down, and that no huge leaps in graphics, as seen in the release of Doom (1993), would ever happen again. This means that a shift in priority could occur, with AI being one of the top contenders. Smith et al. [2] and Serafim et al. [14] believe that a limitation in CPU power and resources is one of the causes for the slow progression of AI development and consequently AI development in games. With relation to this, Skinner and Walmsley [1] outline that powerful GPUs are now handling the load presented by consistently increasing levels of graphics, which frees up the CPU resources to focus on utilizing more powerful AI systems [14]. Serafim et al. [14] also credit a lack of understanding of advanced AI techniques and the uncertainty related to the long-term effects of non-deterministic methods as being causes for the slow progression of AI systems in games. Schaeffer [9] also notes a distinction between academic research, which employs new, complicated AI techniques, and the Games Industry, which typically employs older, simpler methodologies. This distinction is mainly due to having significantly different goals. One such distinction is that, whereas most academic research focuses on creating rational, optimal agents that reason, learn, and react, the industry seeks to build demanding but defeatable opponents that are enjoyable to play against, typically through fully predefined behaviours [15].

2.4.6 Utility of AI Research in Video Games

Presently, video games are being recognized as an ideal environment to test different AI algorithms and techniques because of their complexity, nondeterminism, and limited input [16]. Robertson and Watson [17] outline that video games are a convenient medium as they allow for an unbiased comparison between various algorithms and can be executed as fast as a thousandth of the time it would take for real time, physical tests. The authors go on to explain important research trends in recent years. First of which is General Video Game Playing, which is the development of models capable of learning a large variety of games. Another is Procedural Content Generation, where AI algorithms are used to produce game content, or even design the games themselves. Lastly is AI-assisted design tools, which are used to give game designers immediate feedback and suggestions, thus scaffolding the game design process. Laird and Lent [18] also highlight that games present the opportunity for an AI to be developed in a constrained environment following fixed rules, before being applied to more complex real-world challenges. This is especially useful in situations where the real-world use involves expensive equipment, such as in the robotics field [19].

2.5 Dynamic Difficulty Adjustment

2.5.1 Game Adjustment

Game developers iteratively update game systems using feedback from play testing, changing them until the game is balanced. While this approach cannot be automated, directed mathematical analysis can reveal deeper structures and relationships inside a gaming system, which with the right algorithms and approach, can be adjusted while the game is being played [20]. To make these transitions seamless, developers can make use of the phenomenon called "Change Blindness", which is a failure to detect changes when they occur during saccades, blinks, blank screens, movie cuts, and other interruptions [21].

2.5.2 Precautions for DDA

Hunicke [20] explains that in order to adjust a game experience for a given player, without having negative impacts on well-balanced systems, it is crucial to identify and understand the systems which make it fun and how these can be adjusted to heighten this enjoyment. The author also stresses that adjusting difficulty during a play session, could result in the player feeling cheated. This occurs when adjustments disrupt or degrade the core player experience. The MDA (Mechanics, Dynamics, and Aesthetics) will vary between games and genres, but must be considered and factored into DDA. Mechanics are the different player states and actions involved within the game. Dynamics refers to the variation in difficulty and rewards gained over time within a game. Aesthetics of a game is highly influenced by its Mechanics and Dynamics. In many cases, player expectations according to genre conventions will factor into creating the overarching aesthetics of a video game [20]. Taking the correct measurements was also highlighted within the study, such as taking damage reading over battle and not over the whole playing time, as to retrieve a relevant damage reading with relation to battle.

2.5.3 First Person Shooter Game with Dynamic Difficulty Adjustment

Hunicke [20] used the MDA framework as a guide in order to create a system that adjusts negative feedback without changing the basic FPS genre experience. The developed system controls the game's main inventory Mechanics (health, ammunition, shielding, and weapons), which effect the game's primary exploration and combat Dynamics, while also preserving the overall cycle of activities, which in-turn maintains the game's fast-paced shooter Aesthetics. Hunicke [20] used damage taken, health, mean and standard deviation of current damage rates, and time, to estimate the probability of death in each encounter, which helps to give an indication whether intervention through difficulty adjustment is required. Hunicke [20] stresses the importance of having Adjustment Goals within any DDA controller. The author mentions three different policies, first of which is the Comfort policy, which aims to keep the players reasonably safe. The Discomfort policy instead is optimized to challenge players by limiting supplies and increasing the challenge when the player reaches a predefined range of health. The author continues by describing the Training policy as initially comforting the player, and gradually increasing the discomfort over the course of a level or session. Hunicke [20] implemented a Comfort policy, having a set threshold of 40% for the player's probability of death, which once exceeded, results in difficulty adjustments, which added 15 health points every 100 ticks.

2.5.4 Hypercasual Endless Game with Dynamic Difficulty Adjustment

Yang and Sun [22] explain that in the case of hypercasual endless games, DDA can help to keep the player engaged longer. The research shows that, while initially, most of these games are successful in effectively captivating the attention of the player, most lose the player's interest during the first 7-days of play. Yang and Sun [22] argue that this is due to boredom or frustration brought about by an imbalance between the game's difficulty and the player's skill. Using an effective DDA system, the game can stay within the "flow", which is when the player experiences the ideal challenge and abilities balance. Achieving and maintaining the player's flow during the tutorial or introduction encourages the player to enter the "core loop". The research by Yang and Sun [22] uses a hypercasual endless game which uses a Quick Progressive Difficulty (QPD) system and creates another version which implements DDA instead. The QPD works in such a way that difficulty only rises, and so once a player plateaus, frustration and anxiety start becoming a problem. The research hypothesizes that tackling these elevated, negative emotions through DDA could help keep the player in the "flow", allowing the player to interact longer. The normal version of the game starts the player with 5 lives and lets the player go up a level, therefore going up in difficulty, after gaining 5 consecutive points without losing a life. After this level up, the user is then stuck at this difficulty, or harder if he/she can again gain another 5 consecutive points without losing a life. The adapted version which implements

DDA, allows the player to level up in the same manner, however, adds a condition that allows the player to level down to mitigate the frustration caused by the extreme difficulty. After an increase in difficulty, if the player loses a life, this indicates that the difficulty may be out of sync with the player's abilities at that point in time, which causes a decrease in difficulty by leveling the player down.

2.5.5 Pac-Man with Dynamic Difficulty Adjustment

Hao et al. [23] implemented DDA for the game Pac-Man. This implementation requires that the 'ghosts' are managed using a Monte Carlo Tree Search (MCTS), which is an improved version of the Monte Carlo Simulation algorithm. The 'ghosts' engage in battles against a computer-simulated 'Pac-Man' that follows a predetermined strategy. At each decision point in the maze, the MCTS simulation constructs a search tree encompassing all the permissible moves for the 'ghosts'. The available legal moves include Right, Left, Up, and Down, while any moves that would result in the 'ghosts' or 'Pac-Man' hitting a wall or colliding with each other are eliminated. Hao et al. [23] states that, normally, the longer the simulation time, the more accurate the algorithm's predictions are. Therefore, Hao et al. [23] hypothesized that by adjusting the simulation time, a difference in performance could be obtained. Therefore, the study attempted to implement a DDA system through the manipulation of the 'ghosts' simulation time. This adjustment is centred around the opponents' level of intelligence, ensuring that players engage with adversaries possessing similar capabilities, thereby promoting fairness. It also provides a means of ongoing and adaptable adjustment of the game's difficulty. Lastly, DDA based on MCTS requires extensive computation, however, minimizes the need for domain-specific knowledge and human involvement.

2.5.6 Arcade Shooter Game with Dynamic Difficulty Adjustment

In the study by Wang and Tseng. [24], Artificial Neural Networks (ANNs) were used to control an arcade shooter game's non-player characters (NPCs). The game, which was developed using the XNA game development platform, consisted of having two players shooting fireballs at each other while dodging any incoming attacks. A health and energy system were set in place, after which three ANNs were trained to manage NPCs for three different difficulty settings.

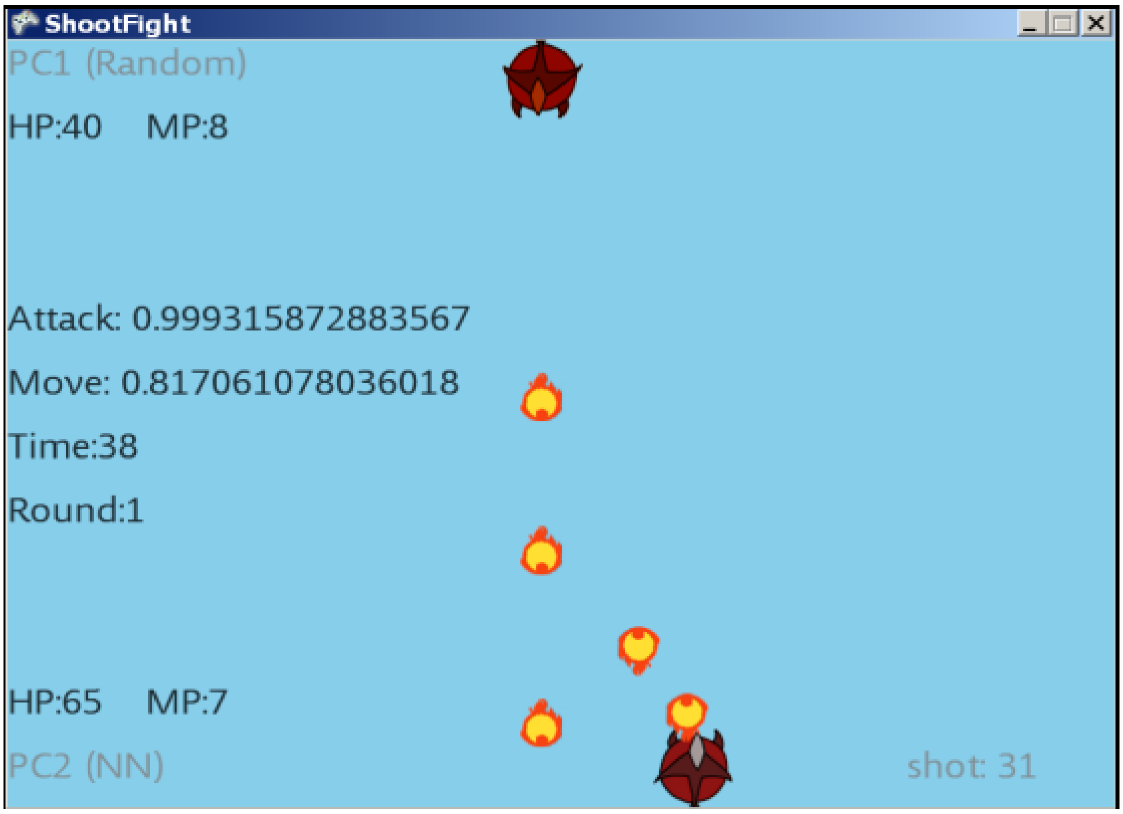


Figure 2.1: *Arcade Shooter Game layout.*

The individual ANN's were trained using specific data sets, derived from players of three different levels of skill (expert, medium, and beginner). To determine the optimal number of neurons in the hidden layer and the appropriate duration for training, a three-fold cross-validation method was employed, which was used to estimate the skill of the model on unseen data. Additionally, the study includes a comparison between the ANN approach and two traditional game AI techniques: finite state machine (FSM) and computer random controlled methods. After embedding the trained ANN into the game, to manage the NPCs' behaviour, Fuzzy DDA was developed. The relative difference of players health points (RDHP) is used as the input for a fuzzy logic system, which uses a membership function to derive a percentage chance for the game's difficulty to be set to either hard, medium, or easy.

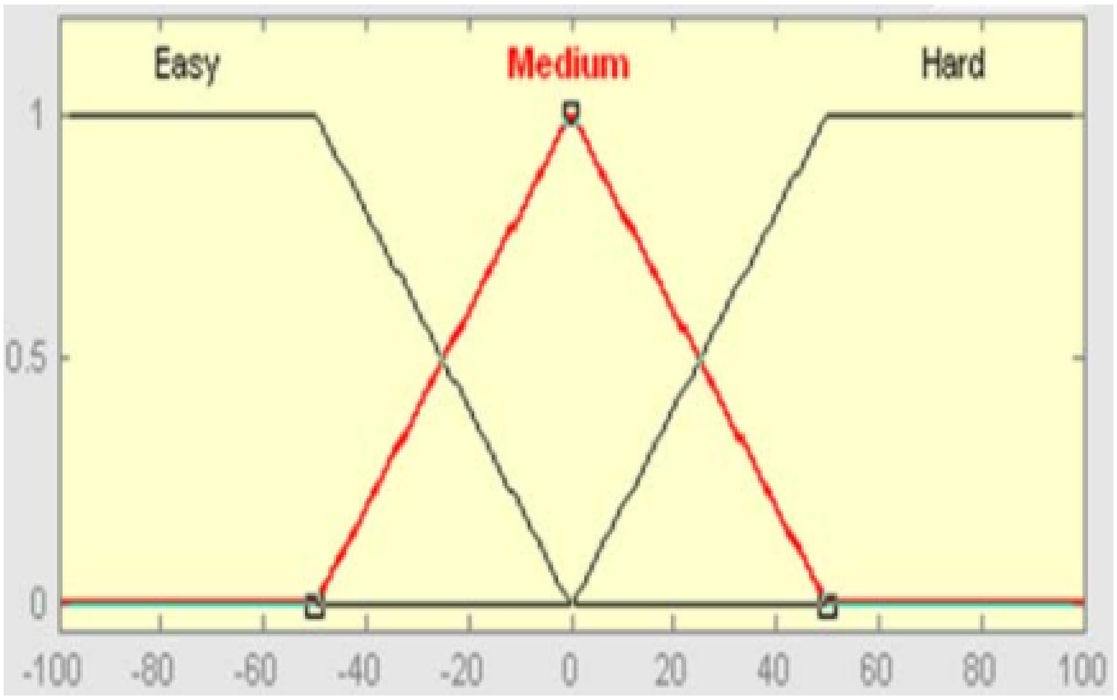


Figure 2.2: *Fuzzy Logic System based on RDHP.*

According to the used fuzzy membership function in [24], when the comparative difference of player's HP is 20, the DDA system had a 40% probability of setting a hard level difficulty and a 60% probability of setting a medium level difficulty.

2.5.7 Experiments

Hunicke [20] initiated the experiment by giving subjects a description of the game and its controls, followed by at least 15 minutes of playtime. Performance data was stored for later revision and subjects were then requested to fill out an evaluation form. The experiment was single-blind, with half of the games adjusted and the other half control [20]. Adversely, Yang and Sun [22] requested that participants first try one version of the game, before proceeding to try the other. This allowed the research to distinguish between the results obtained from each participant for both versions. Scores and frustration evaluations were gathered from each participant after each play session. Hao et al. [23] simulated 'ghosts' controlled by MCTS, against the 'Pac-Man', which could only choose to move forward or turn right at decision points. For each simulation time configuration, a total of 250 gameplays were conducted. Every 50 gameplays, the win-rate acquired by the 'ghosts' was calculated, such that 5 win-rates were extracted from one simulation time. These were then used to calculate an average win-rate, which was plotted in a graph against its respective simulation time.

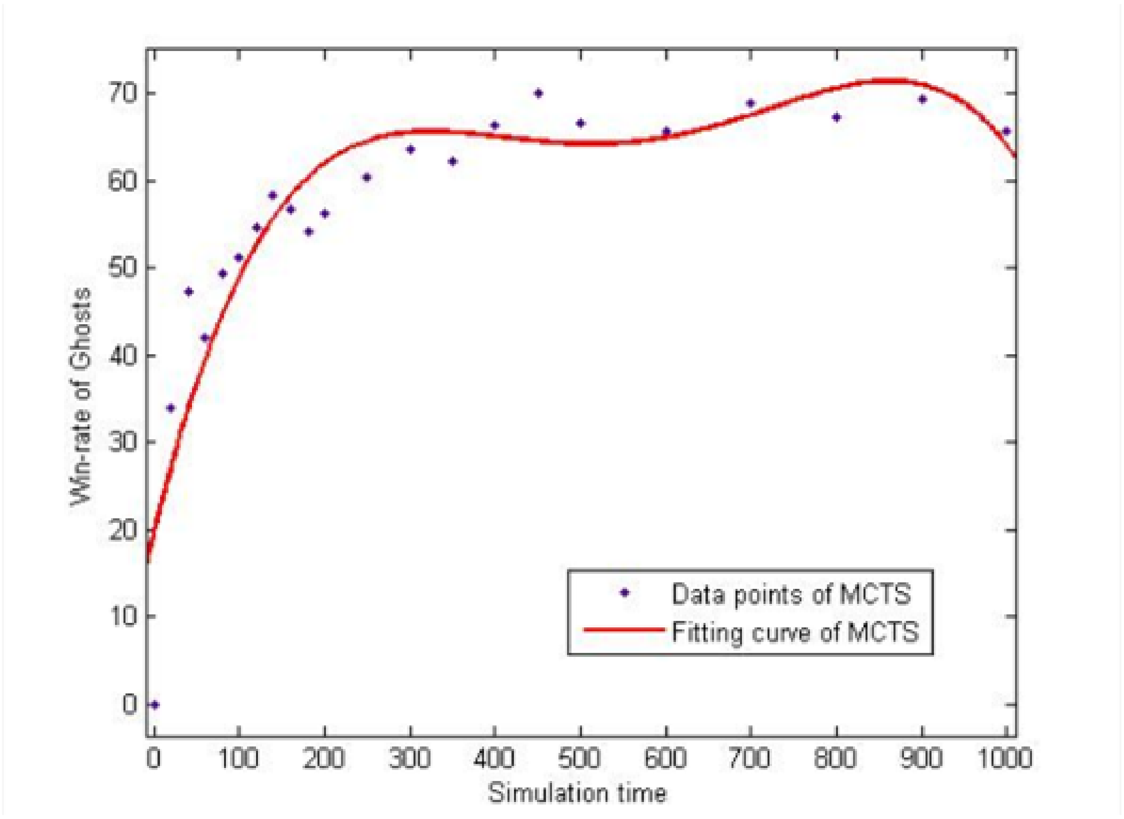


Figure 2.3: *Graph showing the relationship between the MCTS Simulation Time and the Win-Rate for 'Ghosts'.*

In Wang and Tseng [24], the study compares the selected algorithms by pitting them against each other in the developed arcade shooter game. Each variation of the ANN was pitted against FSM and “random control” for 10 rounds each. The relative difference of players health points for each round was noted and plotted on a graph against the respective round number. A positive score indicates that the ANN performed better, whilst a negative score indicates an inferior performance.

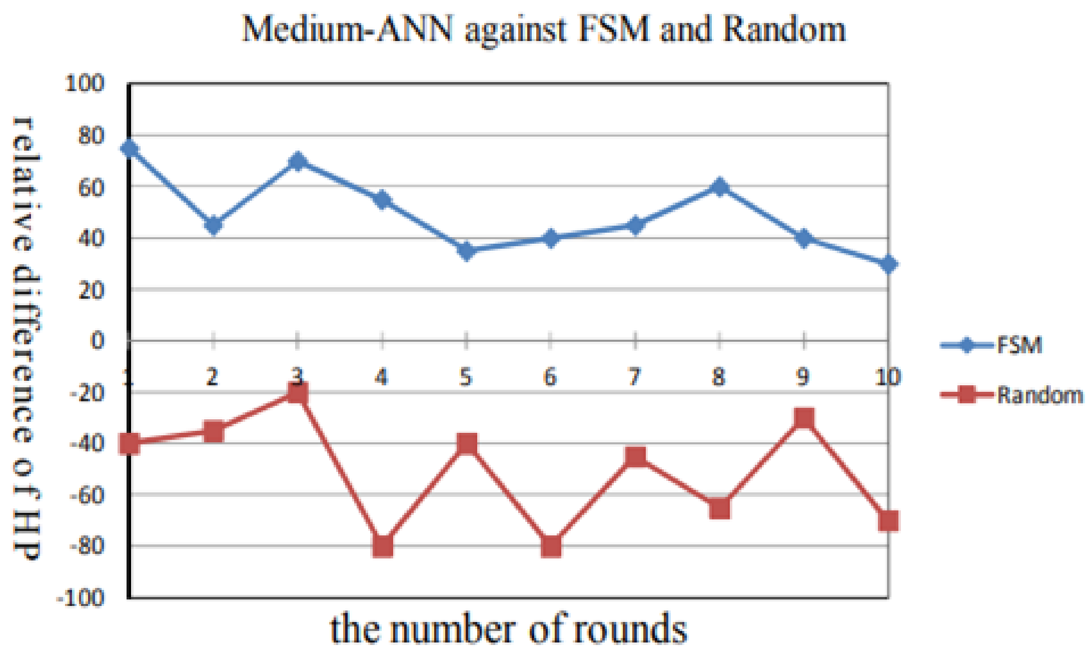


Figure 2.4: *Medium-ANN vs FSM and Random Control.*

This process was then repeated to compare the Fuzzy-DDA-ANN with the aforementioned algorithms. The results obtained were then compared to the results gathered from the previous confrontations which presented promising data.

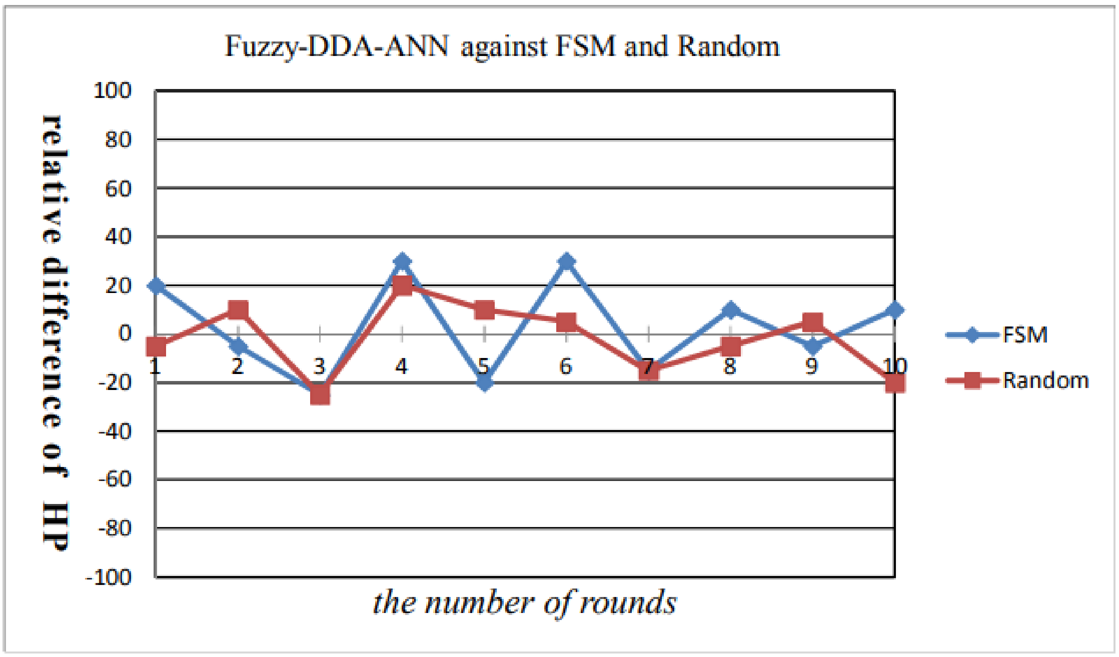
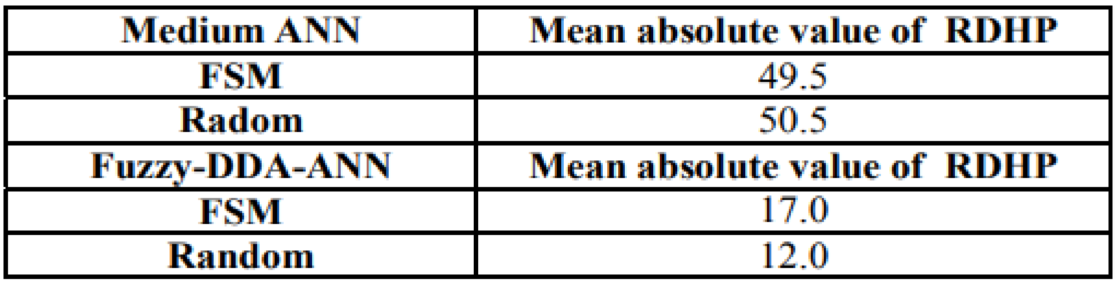


Figure 2.5: *Fuzzy-DDA-ANN vs FSM and Random Control.*

2.5.8 Results

The experiment conducted by Hunicke [20] resulted in a significant decrease in player deaths, with post-play evaluations revealing no differences in enjoyment for novice players, whereas expert players reported slightly elevated levels of engagement. The study also highlights that no significant correlation between adjustment and difficulty evaluation was noted by the subjects. "Change Blindness" was successfully implemented as the attention of players was directed elsewhere. The results of the experiment conducted by Yang and Sun [22] were also promising. For players of all skill types, the DDA system encouraged higher scores, and while casual players also reported a decrease in frustration, experienced players contrastingly reported an increase in frustration. Yang and Sun [22] show confidence that with further research, this too can be improved. Whilst the study by Hunicke [20] implemented DDA through a change in Mechanics, Yang and Sun [22] added DDA by modifying the game's Dynamics, therefore changing its structure. Both approaches are correct, however, the implementation in [22] leaves a larger footprint on the game's Aesthetics. Hunicke's [20] approach maintains the game's format, however risks becoming a nuisance for eagle eyed players which notice the adjustment. Yang and Sun's [22] approach does not attempt to stay anonymous and therefore, does not risk the user feeling cheated by the game. Instead, DDA is implemented such that it is an integral part of the game. Yang and Sun [22] explain that a disadvantage to this is, that it could lessen the enjoyment of the game and heavily change the experience provided by the original. An increase in "flow" was achievable through the implementation presented by Hao et al. [23]. Normal players were found to prefer an even match, which could be achieved by applying the simulation time that achieved a 50% win-rate. Similarly, expert players were found to prefer a more challenging game, which could be satisfied by applying the simulation time which achieved a 70% win-rate. The same could be realised for novice players by applying the simulation time which resulted in a 30% win-rate. With a performance curve which ranges from a 20% to a 70% win-rate, DDA could be adequately implemented by adjusting the simulation time for a MCTS algorithm. However, this technique requires intensive computation and considerable consumption of system resources. Therefore, this application as yet, can only be applied to locally played PC games. Wang and Tseng's [24] initial experiment of pitting the developed ANNs against the FSM and "random control" algorithms gave back valuable data. This comparison found that the ANN trained on the expert player data managed to outperform both other techniques, whereas the ANN trained on the medium player data managed to consistently beat FSM, while regularly losing to the computer random controlled AI. Later, the results obtained from pitting the Fuzzy-DDA-ANN against the same aforementioned algorithms, achieved much less distinct results in terms of the RDHP. When compared with the results obtained when using the medium ANN, it is clear that the Fuzzy-DDA-ANN provides a marginally better balance for opponents of different skills.



Medium ANN	Mean absolute value of RDHP
FSM	49.5
Radom	50.5
Fuzzy-DDA-ANN	Mean absolute value of RDHP
FSM	17.0
Random	12.0

Figure 2.6: *The mean absolute value of RDHP: Medium ANN vs Fuzzy-DDA-ANN.*

2.6 Conclusion

Within this comprehensive literature review, the history of AI and Video Games were briefly outlined to give context to this study. DDA was explained as a concept, and relevant past research were highlighted and explained, through which different methods of implementing DDA were explored. These implementations are used as inspiration and reference for the DDA system developed within this research.

Chapter 3: Research Methodology

3.1 Introduction

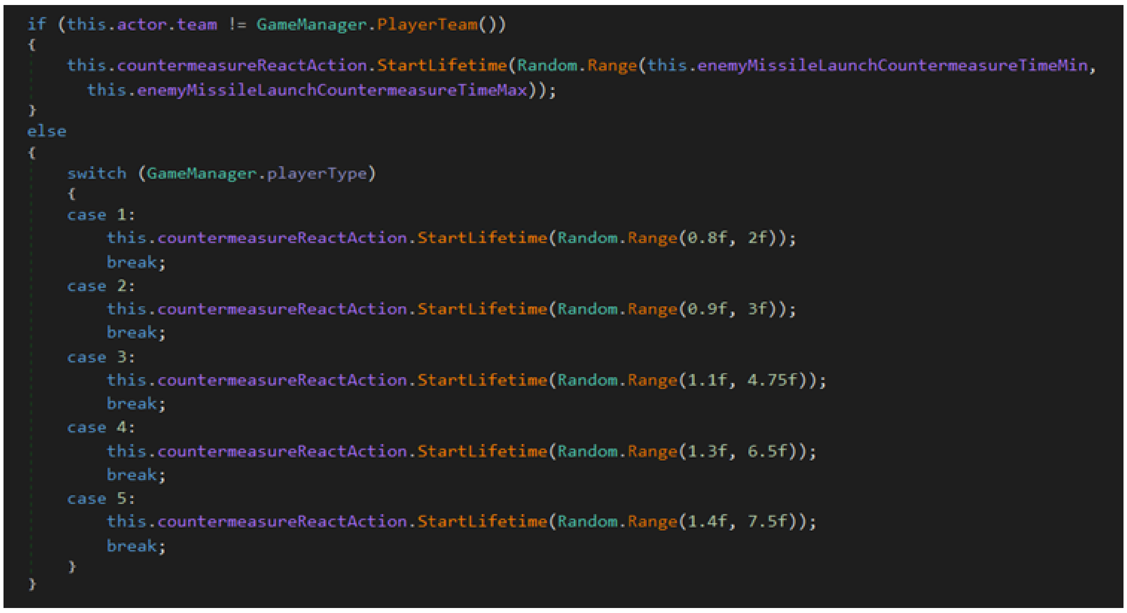
The overarching aim of this study is to gauge the user's appreciation for the inclusion of systems which dynamically adapt a game's difficulty. These changes take place so that specific conditions, set forward by a chosen policy, can be satisfied. As done by Hunicke [20], a Comfort Policy was chosen for this research, where the DDA aims to keep the player relatively safe while keeping the game reasonably challenging.

3.2 'Ravenfield'

The first step was to choose a game to which a DDA AI system could be applied. The First-Person Shooter game 'Ravenfield' was chosen for its accessibility, intrigue, and gameplay balance. 'Ravenfield' was developed by Johan Hassel using the Unity Game Engine. Although not great in number, the game's community is highly active, and is a major driving force for the game's progression. Tools for map creation and feature modding are available to the community to experiment with, and to customise their gameplay. The scope of the game is to defeat the adversary team, through a number of methods which depend on the game mode being played. To carry out this research, the 'Battle' game mode was selected and used, due to its initial complexity for new players. 'Battle' involves having two teams start with 300 unit lives (tickets) and 1 flag each. An additional flag in the centre of the map can be conquered by having team members spend a set amount of time in the flag's vicinity. The team with the most flags captured benefits from an advantage, as 1 ticket is bled from the opposite team every four seconds, for as long as this flag advantage is maintained. The game features a small catalogue of weapons and tools from which the player can create a loadout (inventory). NPCs are given a random loadout of their own, which adds some welcome unpredictability to both team's capabilities. The game also has vehicles such as helicopters which fire high damage projectiles, and other vehicles such as jeeps which offer utility in the larger maps available in the game. For this research, loadout customisation and vehicles were excluded as to contain the complexity, both in scale and power.

3.3 Game Code Changes

Three different versions of the game's executables were used, which fulfilled three different objectives. First of which is the original version, which was used to gather data which could be compared to the data extracted from participants after playing the experimental version. A training version was also developed to be used when training the Deep Q-Network (DQN) model later on. The training version adapted the original version so that the blue team (friendlies) were randomly set to one of five different skill levels at the start of each match. The skill level assigned changed the team's attributes, with skill level 1 being the best, therefore equipping the friendly NPCs with the best set attributes, and with skill level 5 being the worst, therefore equipping the friendly NPCs with the worst set attributes.



```
if (this.actor.team != GameManager.PlayerTeam())
{
    this.countermeasureReactAction.StartLifetime(Random.Range(this.enemyMissileLaunchCountermeasureTimeMin,
      this.enemyMissileLaunchCountermeasureTimeMax));
}
else
{
    switch (GameManager.playerType)
    {
    case 1:
        this.countermeasureReactAction.StartLifetime(Random.Range(0.8f, 2f));
        break;
    case 2:
        this.countermeasureReactAction.StartLifetime(Random.Range(0.9f, 3f));
        break;
    case 3:
        this.countermeasureReactAction.StartLifetime(Random.Range(1.1f, 4.75f));
        break;
    case 4:
        this.countermeasureReactAction.StartLifetime(Random.Range(1.3f, 6.5f));
        break;
    case 5:
        this.countermeasureReactAction.StartLifetime(Random.Range(1.4f, 7.5f));
        break;
    }
}
```

Figure 3.1: *Example code change to enable blue team variation.*

This functionality is the base on which different data was generated during the model training period. These five different skill levels automatically simulated the difference in team performance which during normal play, is caused by the player's performance. This was important in order to train the model, which would need to experience a large amount of unique data in order to learn effectively and thoroughly. Due to the lack of data available, this approach allowed for the generation of versatile data throughout a wide spectrum of skill levels and circumstances. Functionality to log important data was added to the 'BattleMode' class. The data included: tickets remaining for blue team, tickets remaining for red team, flags controlled by blue team, flags controlled by red team and current enemy skill level, as set by the AI model. This data was used in real-time to give context to the AI model. This data was logged in 'GameState.csv', from where the model could then retrieve the necessary observation data. In both the training and the experiment versions, the ‘GameManager’ class was adapted to retrieve the action selected by the AI model by loading the ’AI_Actions.csv’ file and save this into a static variable. The retrieved action is then referred to within the ‘AiActorController’ class, which uses this data to set the skill level of the enemy team. The skill level pre-sets in this case ranged from 1 to 11, with skill level 1 being the best, therefore equipping the hostile NPCs with the best set attributes, and with skill level 11 being the worst, therefore equipping the hostile NPCs with the worst set attributes. Additional flexibility for the red team in skill level pre-sets was implemented purposefully so that the model has flexibility to overcome and adapt to situations as required. In trying to keep the game as simple as possible, a new, moderately scaled map was created using the game’s in-built map creator. The square map allowed for the spawning of both teams at opposite points of the map, which promotes direct confrontation. At both spawn positions, a flag was placed so that each team owned one flag at the start of the match, while another flag was placed in the exact centre so that the teams converged on it using the game’s readily available pathfinding algorithm, which further imposed direct confrontation.

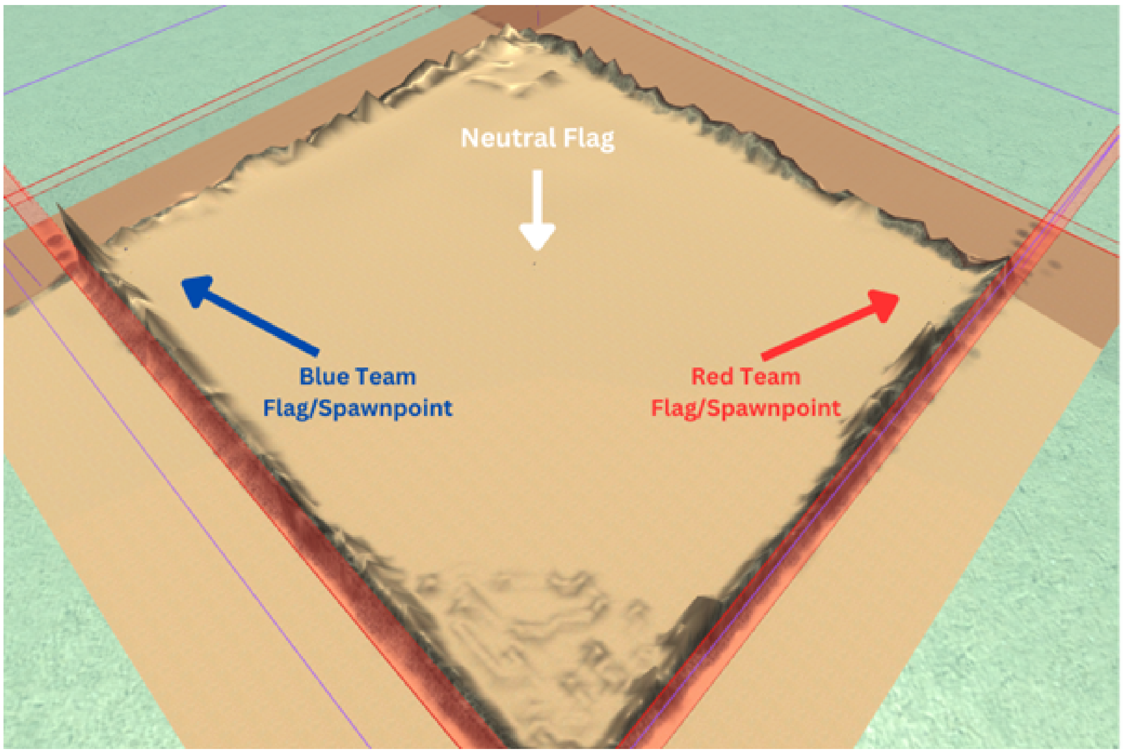


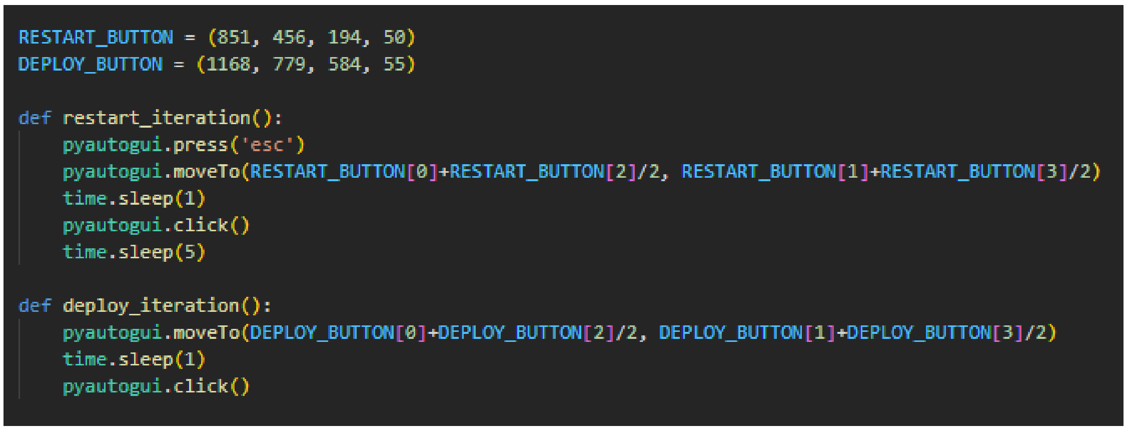
Figure 3.2: *Custom map breakdown.*

3.4 Training Environment Setup

'Ravenfield' and the AI model were set up as individual components because the game files could not be adapted to allow for the integration of a new AI system. To connect these two components, CSV files were used to handle the flow of data. A Jupyter notebook was utilised to run the code necessary in order to implement the required components. The use of a Jupyter notebook allowed for step-by-step execution, which helped in keeping things clean and organised for debugging.

3.4.1 Restart Procedure using ‘PyAutoGui’

‘PyAutoGui’ allowed for automating all button presses and clicks required to restart a ‘Ravenfield’ battle. ‘RESTART_BUTTON’ and ‘DEPLOY_BUTTON’ presets were created, which stored four attributes: start of button from left, start of button from top, width, and height. Using a ruler, the width and height of the on-screen image were recorded, which together with the screen resolution, were used to find different points of the screen through the formula: $\frac{\text{measurement of full screen}}{\text{resolution}} \times$ on-screen measurement related to the button. Example, to find the start of the restart button from the left: $\frac{40\text{cm}}{1920\text{p}} \times 16\text{cm}$ Similarly, an example of finding the size of the button’s height: $\frac{25\text{cm}}{1080\text{p}} \times 2\text{cm}$. Two functions were developed to handle the restart process from start to finish. The first process starts by autonomously inputting the ‘escape’ button, at which point an in-game menu pops up. The mouse is then guided to the set location where the ‘RESTART_BUTTON’ is located, and after a 1 second interval, an autonomous left click starts the game’s restart procedure. Finally, the first function waits 5 seconds before concluding. The second function starts by guiding the mouse onto the ‘DEPLOY_BUTTON’ and after a 1 second interval, it executes a left click which initialises a new ‘Battle’.



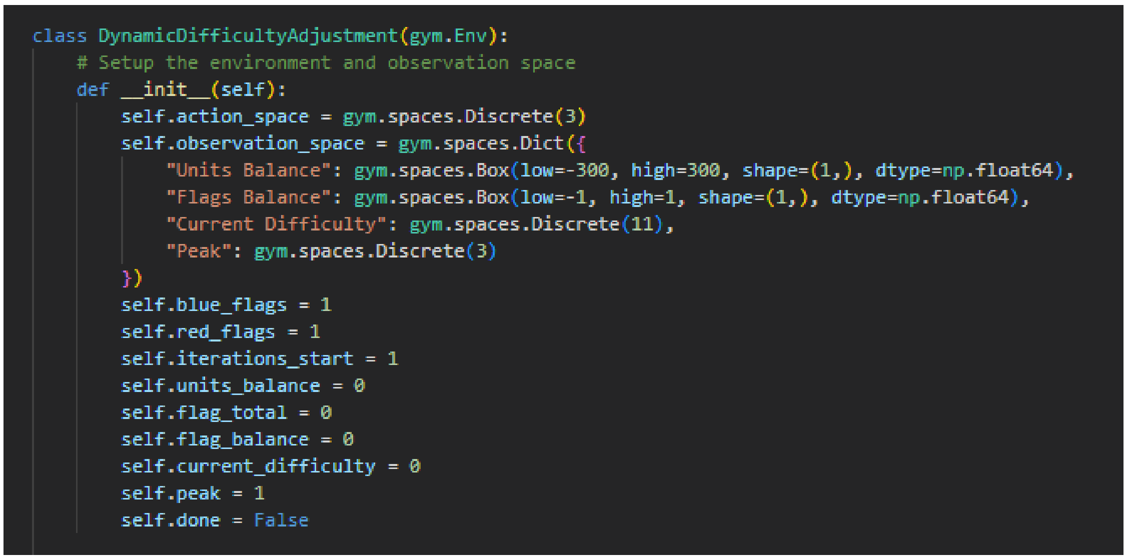
```
RESTART_BUTTON = (851, 456, 194, 50)
DEPLOY_BUTTON = (1168, 779, 584, 55)

def restart_iteration():
    pyautogui.press('esc')
    pyautogui.moveTo(RESTART_BUTTON[0]+RESTART_BUTTON[2]/2, RESTART_BUTTON[1]+RESTART_BUTTON[3]/2)
    time.sleep(1)
    pyautogui.click()
    time.sleep(5)

def deploy_iteration():
    pyautogui.moveTo(DEPLOY_BUTTON[0]+DEPLOY_BUTTON[2]/2, DEPLOY_BUTTON[1]+DEPLOY_BUTTON[3]/2)
    time.sleep(1)
    pyautogui.click()
```

Figure 3.3: *Automation methods setup.*

3.4.2 'OPENAI' Gym Environment

init(self)

A gym environment was then built with five functions. Firstly, the function '_init_' handles the initial shaping of the environment by setting up crucial components, such as the environment's action space and observation space. 0, 1, and 2 where set as the possible values for the action space, respectively representing a decrease, no change, and an increase in game difficulty. The observation space was created as a dictionary which held a value between -300 and 300 representing unit balance, a value between -1 and 1 representing flag balance, a value between 0 and 10 representing the current set difficulty, and a value between 0 and 2 representing peaks in difficulty. In the initialising function, other variables which play important roles within the other functions, are declared, and initialised by a default value.

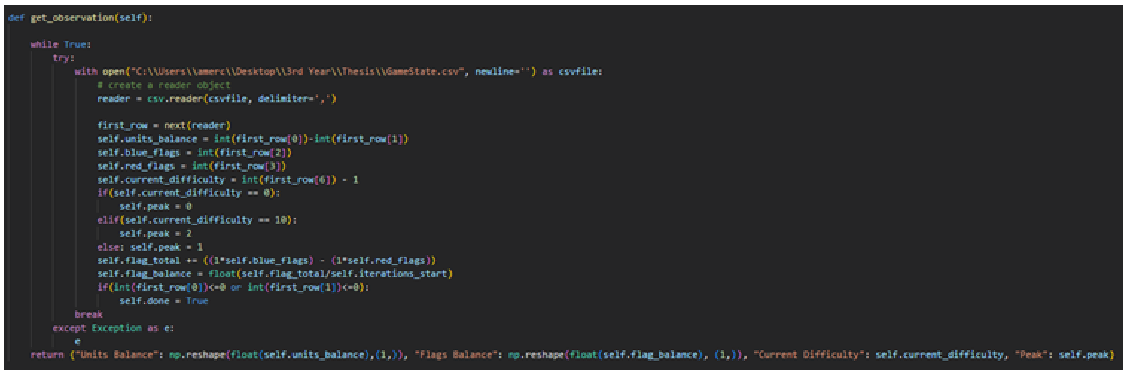


```python
class DynamicDifficultyAdjustment(gym.Env):
    # Setup the environment and observation space
    def __init__(self):
        self.action_space = gym.spaces.Discrete(3)
        self.observation_space = gym.spaces.Dict({
            "Units Balance": gym.spaces.Box(low=-300, high=300, shape=(1,), dtype=np.float64),
            "Flags Balance": gym.spaces.Box(low=-1, high=1, shape=(1,), dtype=np.float64),
            "Current Difficulty": gym.spaces.Discrete(11),
            "Peak": gym.spaces.Discrete(3)
        })
        self.blue_flags = 1
        self.red_flags = 1
        self.iterations_start = 1
        self.units_balance = 0
        self.flag_total = 0
        self.flag_balance = 0
        self.current_difficulty = 0
        self.peak = 1
        self.done = False
```

Figure 3.4: *'_init_' method declaration.*

get_observation(self)

Another function named 'get_observation', handles data retrieval and assignment. Firstly, an infinite loop encompasses everything, inside which a try/catch block is utilised to attempt file reading and value assigning. On a successful attempt, the loop breaks, and the function moves to return the acquired values. Unit balance is retrieved by subtracting the red tickets remaining from the blue tickets remaining, where a negative value means that the blue team are disadvantaged and a positive value meaning the contrary. The current set difficulty is retrieved and subtracted by 1 in order to be refactored to the range accepted by the observation space. The peak is set to 0 when the current difficulty is at its hardest (0) and set to 2 when the current difficulty is set to its lowest (10). Otherwise, the peak is set to 1. The flag balance is calculated by adding the difference in currently owned flags to the total flag balance, and then dividing by the total steps taken since the battle started. Another variable signals whether or not the match should be restarted, depending on whether both teams have units remaining or not. Finally, this function returns a dictionary type, with the needed values to satisfy the environment's observation space.

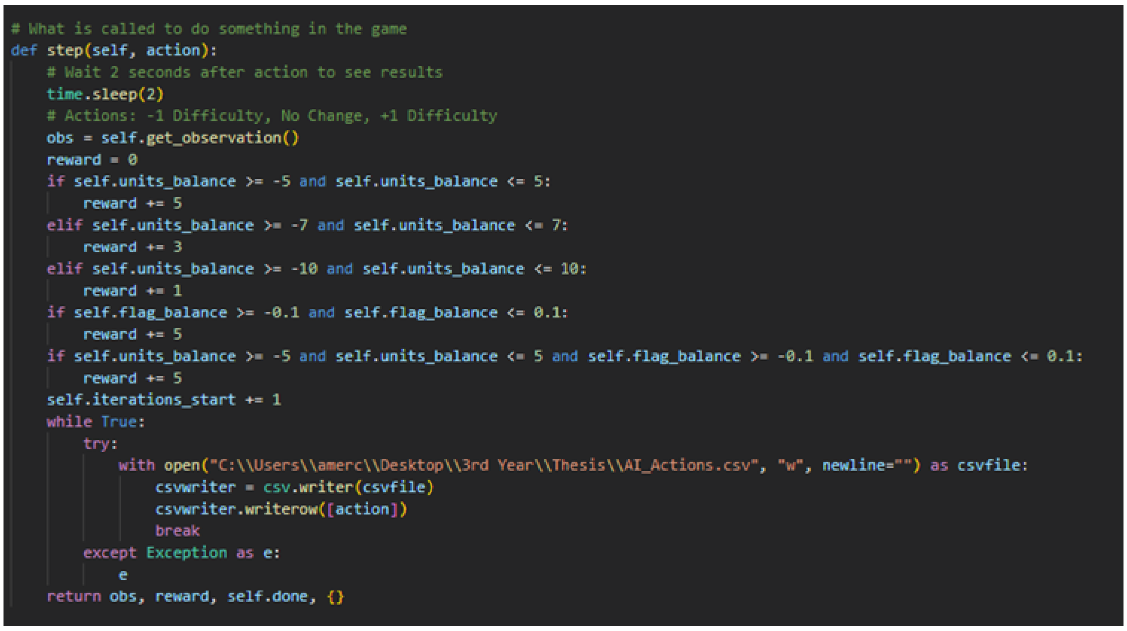


```
def get_observation(self):

    while True:
        try:
            with open("C:\\Users\\amerc\\Desktop\\3rd Year\\Thesis\\GameState.csv", newline='') as csvfile:
                # create a reader object
                reader = csv.reader(csvfile, delimiter=',')

                first_row = next(reader)
                self.units_balance = int(first_row[0])-int(first_row[1])
                self.blue_flags = int(first_row[2])
                self.red_flags = int(first_row[3])
                self.current_difficulty = int(first_row[6]) - 1
                if(self.current_difficulty == 0):
                    self.peak = 0
                elif(self.current_difficulty == 10):
                    self.peak = 2
                else: self.peak = 1
                self.flag_total += ((1*self.blue_flags) - (1*self.red_flags))
                self.flag_balance = float(self.flag_total/self.iterations_start)
                if(int(first_row[0])<=0 or int(first_row[1])<=0):
                    self.done = True
            break
        except Exception as e:
            e
    return {"Units Balance": np.reshape(float(self.units_balance),(1,)), "Flags Balance": np.reshape(float(self.flag_balance), (1,)), "Current Difficulty": self.current_difficulty, "Peak": self.peak}
```

Figure 3.5: *'get_observation' method declaration.*

step(self, action)

The 'step' function runs with every action taken by the model. It starts by waiting 2 seconds for the action to take effect and produce results within the game, after which the 'get_observation' function is run. Using the retrieved data, the model is rewarded based on its performance. The step counter used to calculate the flag balance in incremented and the episode length, which prompts an environment reset, is decremented. This is followed by an infinite loop that contains a try/catch block which attempts to save the action to 'AI_Actions.csv'. On successfully saving, the loop is broken, and the function returns the observation, the gained reward, and a variable signalling whether the episode is done or not.



```python
# What is called to do something in the game
def step(self, action):
    # Wait 2 seconds after action to see results
    time.sleep(2)
    # Actions: -1 Difficulty, No Change, +1 Difficulty
    obs = self.get_observation()
    reward = 0
    if self.units_balance >= -5 and self.units_balance <= 5:
        reward += 5
    elif self.units_balance >= -7 and self.units_balance <= 7:
        reward += 3
    elif self.units_balance >= -10 and self.units_balance <= 10:
        reward += 1
    if self.flag_balance >= -0.1 and self.flag_balance <= 0.1:
        reward += 5
    if self.units_balance >= -5 and self.units_balance <= 5 and self.flag_balance >= -0.1 and self.flag_balance <= 0.1:
        reward += 5
    self.iterations_start += 1
    while True:
        try:
            with open("C:\\Users\\amerc\\Desktop\\3rd Year\\Thesis\\AI_Actions.csv", "w", newline="") as csvfile:
                csvwriter = csv.writer(csvfile)
                csvwriter.writerow([action])
                break
        except Exception as e:
            e
    return obs, reward, self.done, {}
```

Figure 3.6: *'step' method declaration.*

reset(self)

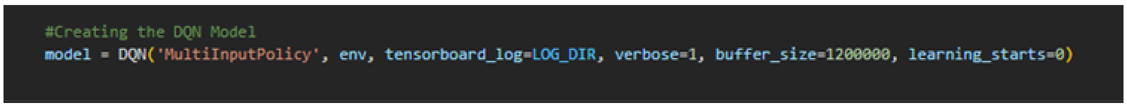
The environment also required a function to facilitate it's reset. This function starts off with a contingency procedure that maximises the 'Ravenfield' window and then waits 2 seconds, therefore making sure that any following button presses are placed within 'Ravenfield'. At this point, the reset function calls the restart and deploy functions declared at the start of the Jupyter notebook, which carry out a match restart within 'Ravenfield'. The episode length is then given a value depending on how many episodes have taken place beforehand. The episode counter is then incremented, and all other variables are assigned their default values. The function then waits 60 seconds to continue, giving the time required for initial collision between the teams, such that no rewards are given to the model for inconsequential actions. Finally, the 'GameState.csv' is emptied and an observation is returned using the 'get_observation'.

render(self)

The final function included was the empty render function, which is necessarily declared for the environment to be valid and to fully implement the gym interface.

3.4.3 Dynamic Difficulty Adjustment Model Creation

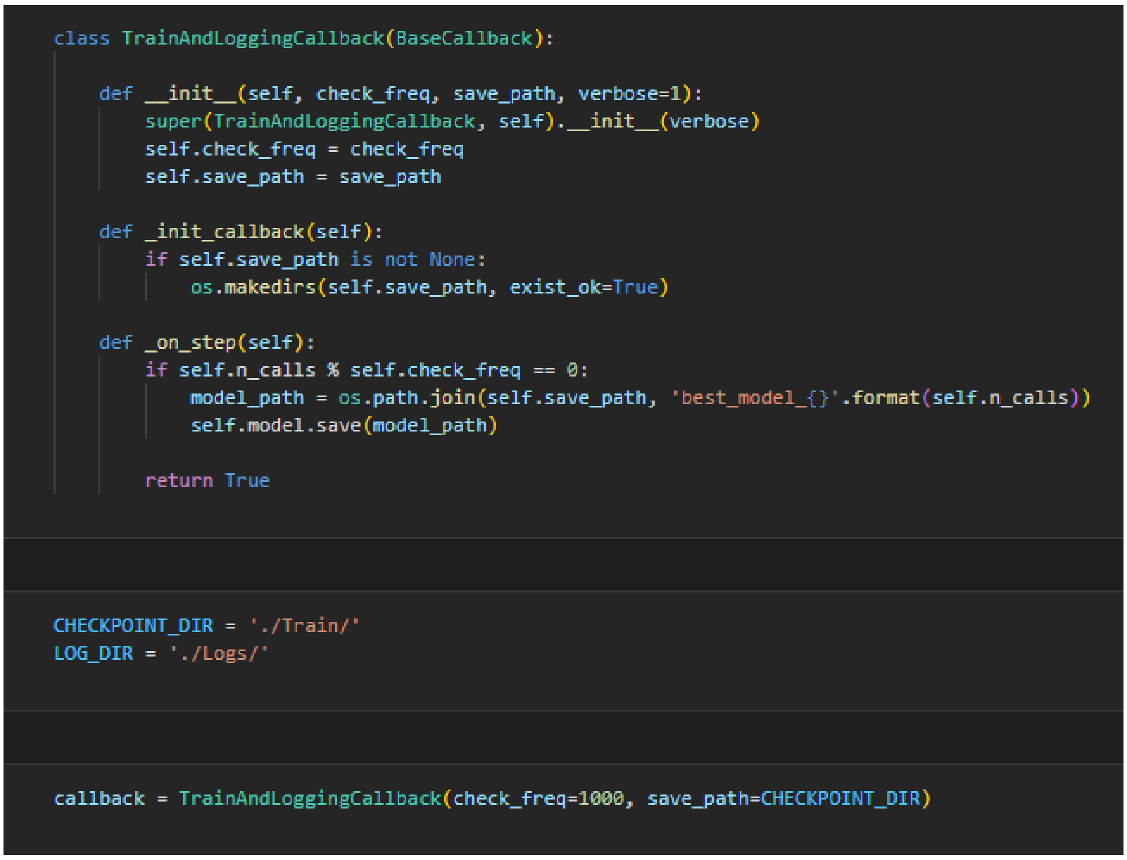
The AI model was created using the ’stablebaselines3’ library available for Python. For this study, it was observed that a Multi Input Policy setting would be ideal to handle the necessarily complex observations. All models were created with the same parameters as specified in Figure 3.7.



```
#Creating the DQN Model
model = DQN('MultiInputPolicy', env, tensorboard_log=LOG_DIR, verbose=1, buffer_size=1200000, learning_starts=0)
```

Figure 3.7: *Deep Q-Network model initialisation.*

The ‘env’ is an initialised object of the gym environment class created. The ‘tensorboard_log’ sets the directory where logs for TensorBoard, a visualization tool, will be stored. The ’verbose’ parameter controls the verbosity level of the DQN algorithm’s output during training. A value of 1 indicates a moderate level of verbosity, providing some information about the training progress and performance. The ’buffer_size’ parameter specifies the size of the replay buffer used in the DQN algorithm. The replay buffer stores experiences (state, action, reward, next state) for experience replay, which helps to improve learning efficiency. A ‘buffer_size’ of 1,200,000 is capable of storing a substantial number of experiences. The ’learning_starts’ parameter is set to 0, which indicates that training should start from the first step. An observation is the data which the model associates with an action during training. Initially the observation included: the blue team's remaining tickets, the red team's remaining tickets, flag balance and current set difficulty. Later in the research, the final model was trained using slightly modified observations which also included a peak parameter, which showed whether a maximum or minimum peak in set difficulty had been reached. A class that handles logging and saving during training was created and initialised so that the model saved its best version every 1000 steps.



```python
class TrainAndLoggingCallback(BaseCallback):

    def __init__(self, check_freq, save_path, verbose=1):
        super(TrainAndLoggingCallback, self).__init__(verbose)
        self.check_freq = check_freq
        self.save_path = save_path

    def _init_callback(self):
        if self.save_path is not None:
            os.makedirs(self.save_path, exist_ok=True)

    def _on_step(self):
        if self.n_calls % self.check_freq == 0:
            model_path = os.path.join(self.save_path, 'best_model_{}'.format(self.n_calls))
            self.model.save(model_path)

        return True
```

```python
CHECKPOINT_DIR = './Train/'
LOG_DIR = './Logs/'
```

```python
callback = TrainAndLoggingCallback(check_freq=1000, save_path=CHECKPOINT_DIR)
```

Figure 3.8: *The class and subsequent settings responsible for saving the DQN model's progress during training.*

This allowed for detailed comparisons between models in terms of effectiveness and improvement rate. Code was added so that a DQN model could be loaded and tested at different stages of training. Different models could be loaded and compared to other models through testing. The initial model was trained for 265000 steps, which took 5 days of continuous training. The second model was trained for 230000 steps and took 7 days of training. The increase in training time was due to adding a 2 second buffer in the step function of the gym environment, before retrieving the observation. Just as Hunicke [20] implemented a threshold of 40% for the player's probability of death before carrying out difficulty adjustments, this DDA model carried out adjustments based on the teams remaining unit lives and the flag control balance. The created gym environment had several conditions which when satisfied, rewarded the AI model accordingly. The first condition pertains to the difference in team tickets (unit balance), where a difference of 5 tickets or less granted the model a reward with a value of 1. Simultaneously, another reward of the same value was earned by the model upon satisfying another condition, which is having the flag balance of the two teams within 10% of each other. Another reward with a value of 1 was given to the model when the previous two conditions were satisfied concurrently. The final model used the same conditions as the first model, however the reward that was gained for satisfying each of these conditions was heavily magnified to return a value of 5. Another two conditions were added to the gym environment with the aim of encouraging the model to improve recovery and reward maximisation. One of these conditions required that the unit balance difference be 7 or less, at which case the model was rewarded with a value of 3. Otherwise, a minimal reward of value 1 was given when the unit balance difference was 10 or less.

3.5 Gauging User Feedback

3.5.1 Data Collection Method

A Four Phase experiment was planned and conducted, during which valuable data was collected. The data was gathered using a mixed methods approach, which involved participants answering both closed ended questions, as well as open ended questions where DDA could be discussed more openly. A number of questions for each Phase were prepared with the intent of gathering as much supporting data as possible, within the boundaries presented by time. 20 participants were invited and accepted to take part in the experiment. As done by [19], the participants were divided into two groups, which differentiated only in the order of the Phases carried out. Group A carried out Phase 2 followed by Phase 3, whilst Group B carried out Phase 3 before moving on to Phase 2. This was done to mitigate bias from the questions in Phase 4, where a person's answers could have been heavily influenced by the Phase they had last completed. This was also helpful in mitigating the influence of participant growth and adaptability from the results obtained in Phases 2, 3 and 4. A number of questions were prepared for each individual Phase. These questions were the main source for all the primary data collected.

3.5.2 The Four Phase Experiment

Phase 1

The participants were all given an individual explanation of the game and its mechanics. Specifically, the rag dolling effect, the manual reloading, and the loadout selection. Subsequently, the participants were given a brief overview of the controls. It was then explained that following a number of pre-play questions, they would be asked to play a single match, after which further instructions would be given. The participants were then asked to answer the Phase 1 questions, which prompted self-assessment on skills related to the experiment and gauged the participants' expected outcome of their upcoming play session.

Phase 1 Questions

1. How often do you play digital games?
2. How would you rate your own skill in playing First Person Shooter Games? (1-5)
3. How would you rate your adaptability to new game experiences? (1-5)
4. Which of these do prefer the game to be: Easy, Challenging, or Extremely Challenging?
5. Have you ever played 'Ravenfield'? If yes, give an estimation of total play time.

For the first question, participants were asked to tick the time value which they agreed with the most. This question was useful in determining whether a participant is experienced in playing digital video games, which gave insight into the likeliness of the participant adapting and improving during or between Phase 2 and Phase 3. For the next two questions, a rating scale system was used. Therefore, the questions could be answered with a value between 1 and 5, including the thresholds themselves, with 1 being the worse rating and 5 being the best rating. The second question provided an estimation of the participant's skill in similar games, which could later be compared to results logged after play in Phases 2 and 3. The third question gave a further indication as to whether the participant could improve during or between Phase 2 and Phase 3. The fourth question was used to give context to the responses correlating to the preferences, which were given later, during Phase 4. The fifth and final question of Phase 1 gave further context to the answers that were later provided in Phase 4's first question. If the participant had played 'Ravenfield' before, this could have had an impact on whether the participant became aware of the Dynamic Difficulty Adjustment, as he/she may have had become familiar with the game's systems.

Phase 2 and Phase 3

In Phase 2, the participants were instructed to play a match using the unchanged version of the game 'Ravenfield'. They were allowed to use whatever loadout they wanted and were instructed to play until the match finished through either victory or defeat. After the play session, participants were asked to answer questions related to the current Phase. These questions measured the levels of enjoyment, engagement, and difficulty, as experienced by the participant. Phase 3 was exactly the same as Phase 2 with a single and important change. During this phase, the participants played a match using the experimental version of the game 'Ravenfield' which utilised the Dynamic Difficulty Adjustment managed through the AI Model. After the end of the match, evaluation was carried out as done in Phase 2.

Phase 2 and Phase 3 Questions

1. How challenging was the match? (1-5)
2. How engaged were you during play? (1-5)
3. How much would you say you enjoyed it? (1-5)
4. How well-balanced was the game's difficulty? (1-5)
5. Did you notice anything strange during the match?

The first 4 questions used a rating scale system. Therefore, the questions could be answered with a value between 1 and 5, including the thresholds themselves, with 1 being the worse rating and 5 being the best rating. The first question was asked to compare the results between the normal and adjusted gameplay after Phases 2 and 3 were both completed. The second question measured the levels of engagement, which could then be compared to see which between Phases 2 and 3, offered the more interesting experience. The third question's purpose was similar to the previous questions, as it compared the participants' enjoyment for both cases. The fourth question again compared the two system's capabilities in offering the players a well-balanced experience. The final question for Phases 2 and 3, asked the participants whether they noticed anything out of the ordinary during their play. This was done to see if the phenomena of 'Change Blindness', as described by Simons [21], was present during play, such that changes in difficulty were not identified.

Phase 4

The final Phase started by finally explaining the research and the concept of Dynamic Difficulty Adjustment to the participant. Now that they were given clear context, they were asked whether they could identify which of the matches played had Dynamic Difficulty Adjustment. Finally, they were asked other questions related to their subjective opinions regarding the use of Dynamic Difficulty Adjustment in Digital Games.

Phase 4 Questions

1. Could you guess which of the matches played had Dynamic Difficulty Adjustment?
2. Do you feel that the adjustments were suitably balancing the game?
3. Do you feel that the adjustments effected your performance?
4. Did the adjustment effect your enjoyment, frustration, or perception of the game?
5. What are your general thoughts on Dynamic Difficulty Adjustment? Do you think it is Beneficial or Detrimental to the game playing experience?

The first question built on the last question of Phases 2 and 3, and solidified whether or not the adjustment was noticeable. The final three questions were open-ended and encouraged discussion. The answer to these questions directly co-relates to the set research questions, therefore to the purpose of this study.

3.6 The Equipment Used

For all stages of this research, an ASUS ROG STRIX G15 was used with the following specifications:

- Central Processing Unit: Intel I7-11750H
- Memory / RAM: 16GB
- Graphical Processing Unit: Nvidia RTX 2060 (Laptop)

Additionally, during the experiment, a Xiaomi Tab 5 was used to record the statistics at the end of each game played.

3.7 Conclusion

The methodology explained, enables this research to provide results on the subjective experience of 20 participants related to the implementation of DDA explained here, while also gathering feedback and ideas about the ideal use of DDA as seen subjectively by the participants. This will enable this research to compare participant scores and views which may be used to deduce new ways of implementing such a system.

Chapter 4: Analysis of Results and Discussion

The results for this study are split into three main parts. First, the accuracy of the AI model trained will be taken into consideration by comparing rewards obtained by the original game and the experimental game. Next, the results from Phases 1, 2, and 3 are discussed together to compare and highlight the major differences from the quantitative results obtained by the participants after playing both the original and the experimental versions. Subsequently, the results obtained from Phase 4 are discussed, supported by the data gathered in Phase 1, as to provide the best context for the qualitative data gathered during the post-experiment interviews.

4.1 AI Model Review

The results in Figure 4.1 were generated through ’TensorBoard’ during training. This data displays the progression made during the first part of the model’s training. As the duration of the game is increased by the environment, the model initially is not capable of prolonging the game. As seen in the left graph of Figure 4.1, by time, the model became better at prolonging the game through balancing the teams. The right graph shows the progression made by the model in maximising the rewards earned from the training environment.

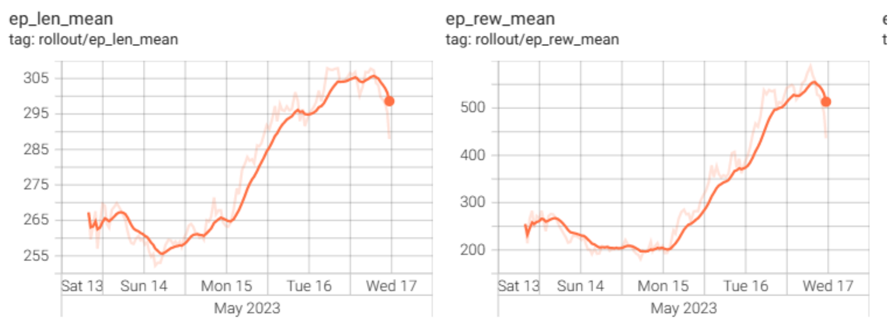


Figure 4.1: *Average episode length (Left) and average reward gained per episode (Right).*

4.2 Questionnaire Results for Group A

This group initially played the original game, after which they played the experimental game. The feedback provided from the questionnaire in Phase 1, helps to give a better understanding of the demographics of the group.

4.2.1 Play Time

With regard to playtime per week, 4 out of the 10 participants have no playtime whatsoever, whilst the range which involved having little or extreme amounts of playtime, represented 1 participant each. The more common ranges represented 2 participants each. As seen in Figure 4.2, the group had a wide and balanced distribution.

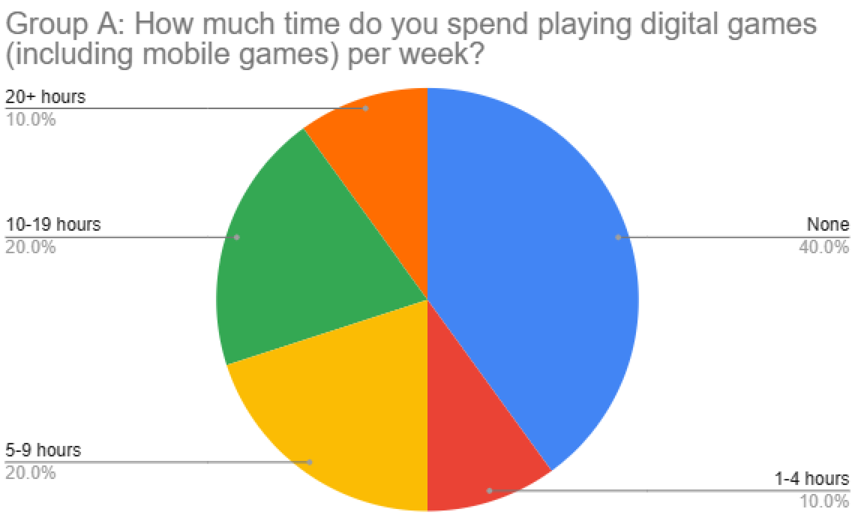


Figure 4.2: *Distribution of Group A's participants' playtime.*

4.2.2 FPS Game Skill Rating

Through self-assessment, an indication of the participants' skill in FPS games was gathered as shown in Figure 4.3. On a scale of 1-5, the most frequent response was the middle ground, having been chosen by 4 participants. The second most frequent response was 1, with three participants being complete novices in this game genre. Another two participants classified themselves as having basic skills by choosing 2. One participant from this group selected the value 4, which signifies an above average skill.

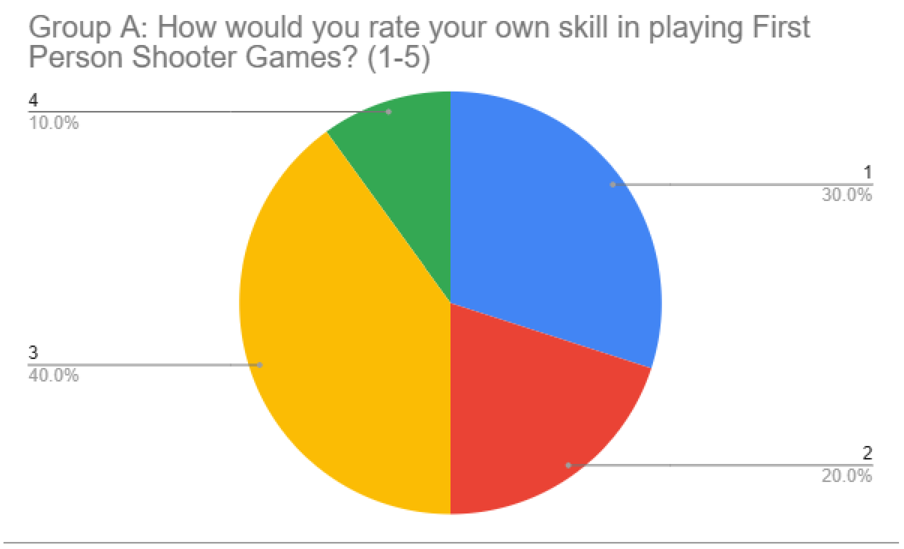


Figure 4.3: *Self-assessments from Group A's participants on their skill level in FPS.*

4.2.3 Adaptability

Figure 4.4 explains the participants' self-assessment regarding their adaptability. On a scale of 1-5, the most common selection was the value 3, with four participants agreeing with having a normal level of adaptability. Three participants agreed with being highly adaptable by selecting the value 4. Another two participants and the remaining participant agreed with the values 1 and 2 respectively. The average adaptability score for Group A was derived to be 2.8 out of 5. Therefore, a noticeable improvement should be visible between the first and second game played.

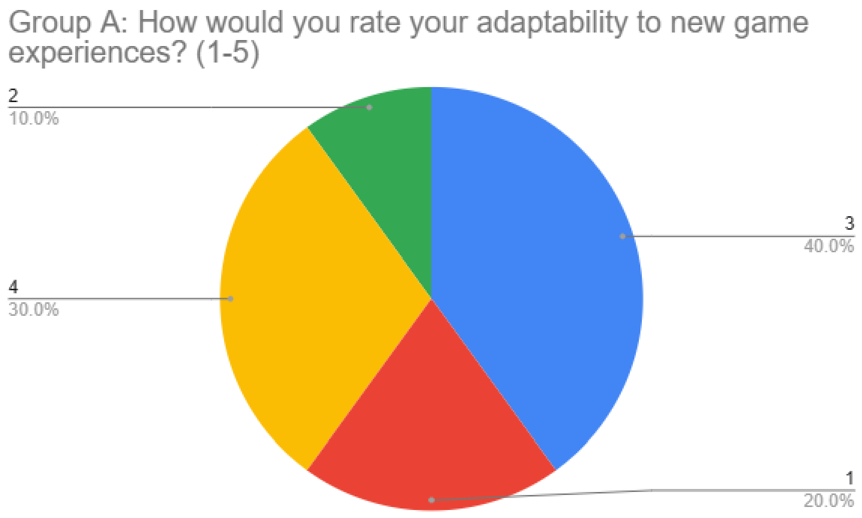


Figure 4.4: *Self-Assessments from Group A's participants on their level of adaptability.*

4.2.4 Game Difficulty Preferences

The data in Figure 4.5 shows that most of the participants prefer a game to feel challenging. The remaining two participants stated they prefer the game to feel easy as this brings more enjoyment. This result is important because it signifies the value of having a well-balanced and challenging difficulty, however it also highlights that no single difficulty setting can appease all audiences.

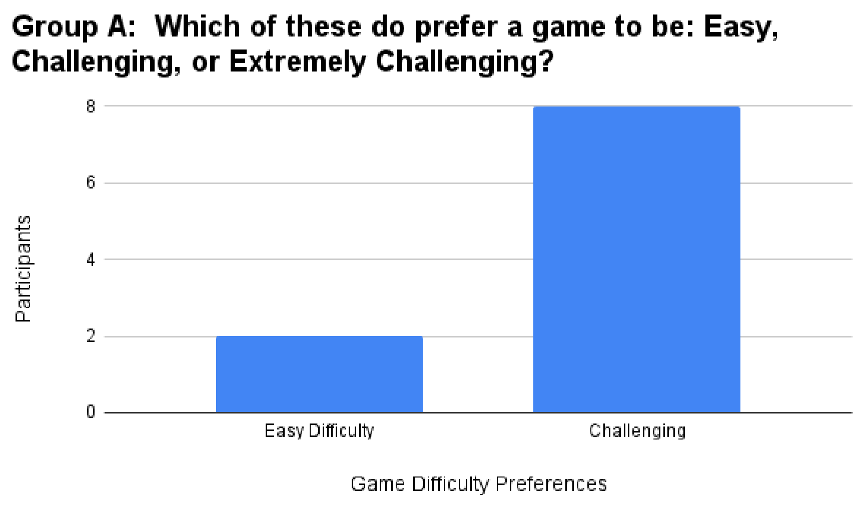


Figure 4.5: *Difficulty preferences for Group A's participants.*

4.2.5 Challenge Scores

Comparing the feedback shown in Figure 4.6 and Figure 4.7, a steep increase in difficulty was felt by the participants between the original and the experimental version. Through the results obtained during Phase 1, it can be deduced that for the majority of the participants, having the average challenge score transition from 2.5 to 3.6, is a step in the right direction.

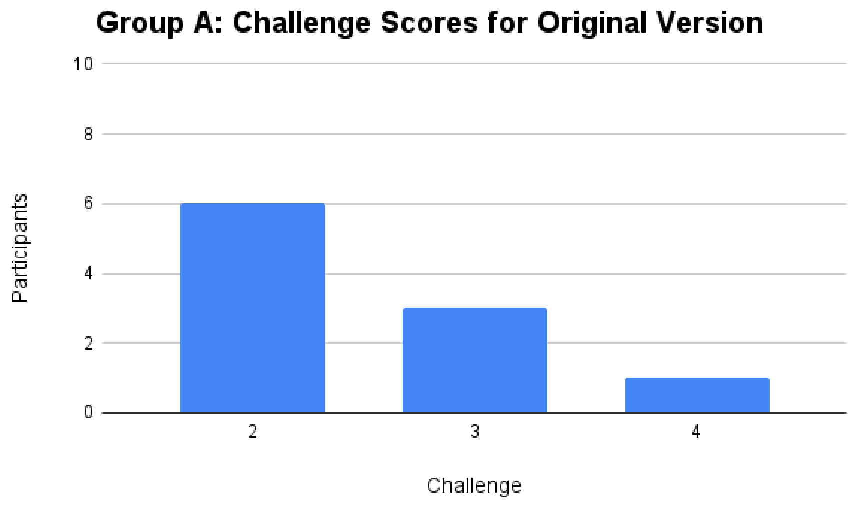


Figure 4.6: *Challenge scores for the original version from Group A's participants.*

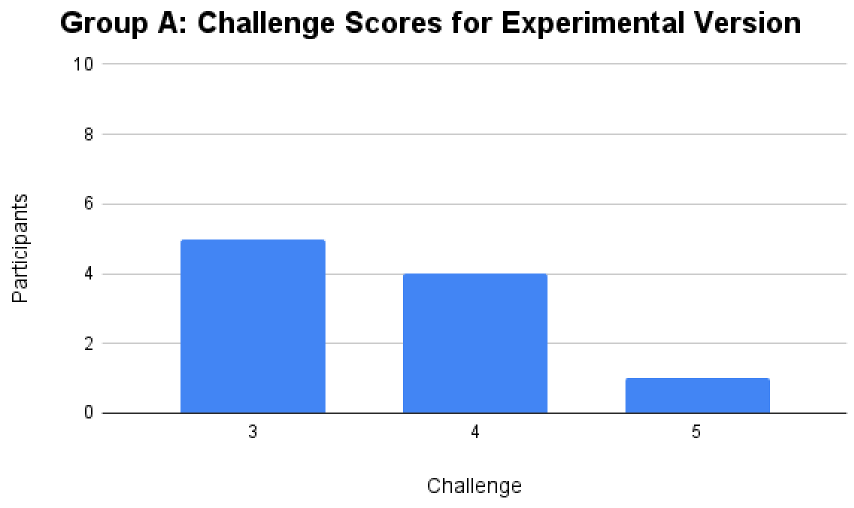


Figure 4.7: *Challenge scores for the experimental version from Group A's participants.*

4.2.6 Engagement Scores

Figure 4.8 and Figure 4.9 show a slight convergence of engagement scores onto the 3-point and 4-point marks. Both scores result in an average engagement of 4.2 out of 5.

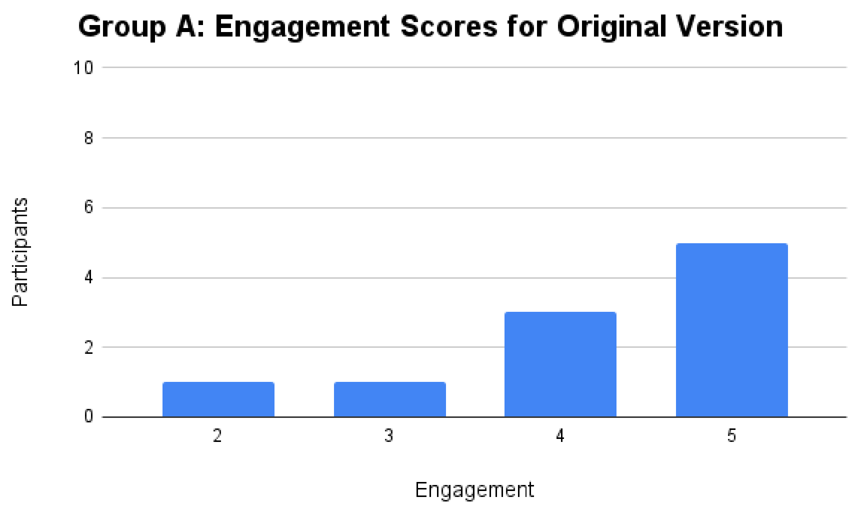


Figure 4.8: *Engagement scores for the original version from Group A's participants.*

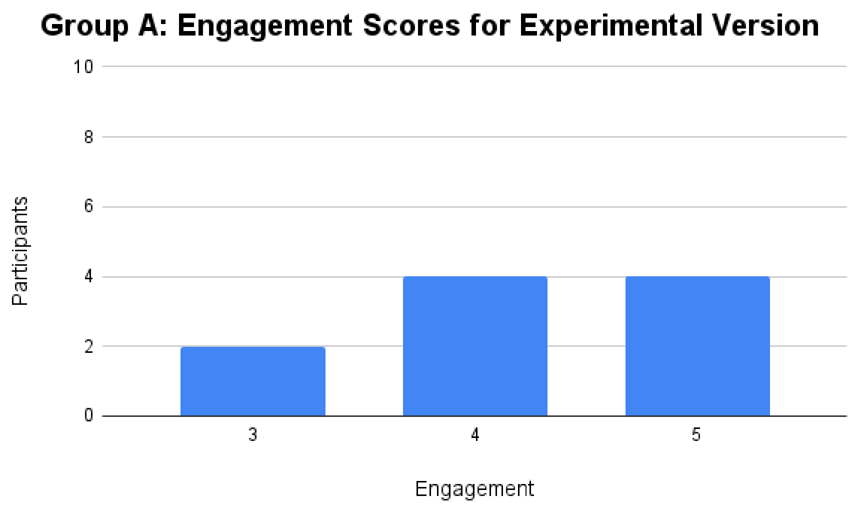


Figure 4.9: *Engagement scores for the experimental version from Group A's participants.*

4.2.7 Enjoyment Scores

A slight elevation in enjoyment was noted as the scores show a 2-point increase and a 1-point decrease concurrently. The average score for the original version in Figure 4.10 was 4.2 out of 5 whereas the average score for the experimental version in Figure 4.11 was 4.3 out of 5.

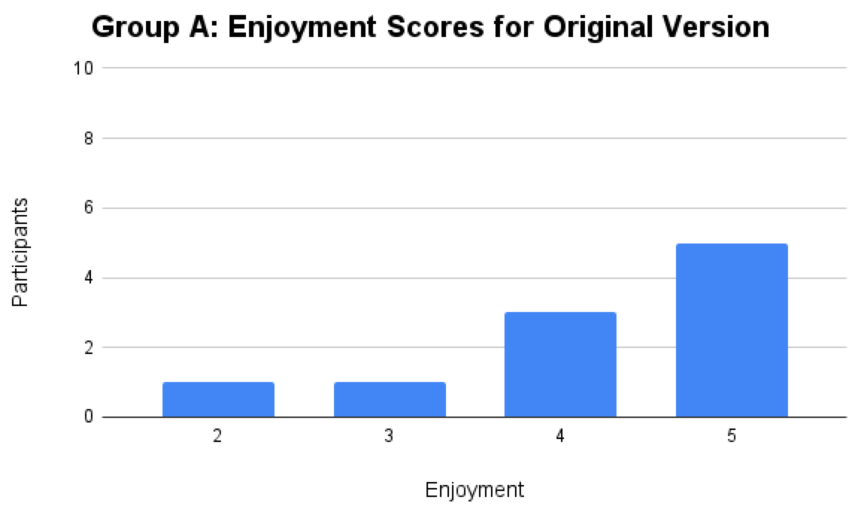


Figure 4.10: *Enjoyment scores for the original version from Group A's participants.*

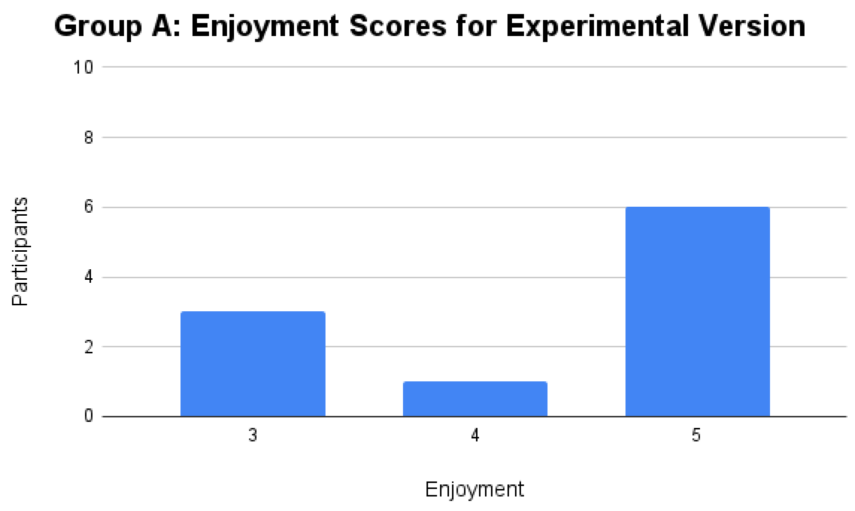


Figure 4.11: *Enjoyment scores for the experimental version from Group A's participants.*

4.2.8 Balance Scores

In the experimental version, participants felt an increase in balance between the teams. As shown in Figure 4.12 and Figure 4.13, there was a significant increase at the 3-point and 4-point marks. The average score for the experimental version was 3.5 out of 5 whilst the average score of the original was just 2.5 out of 5.

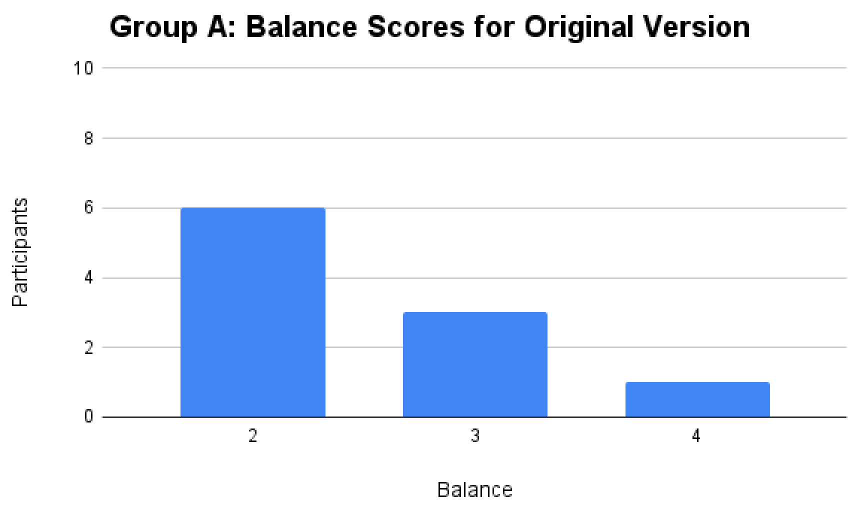


Figure 4.12: *Balance scores for the original version from Group A's participants.*

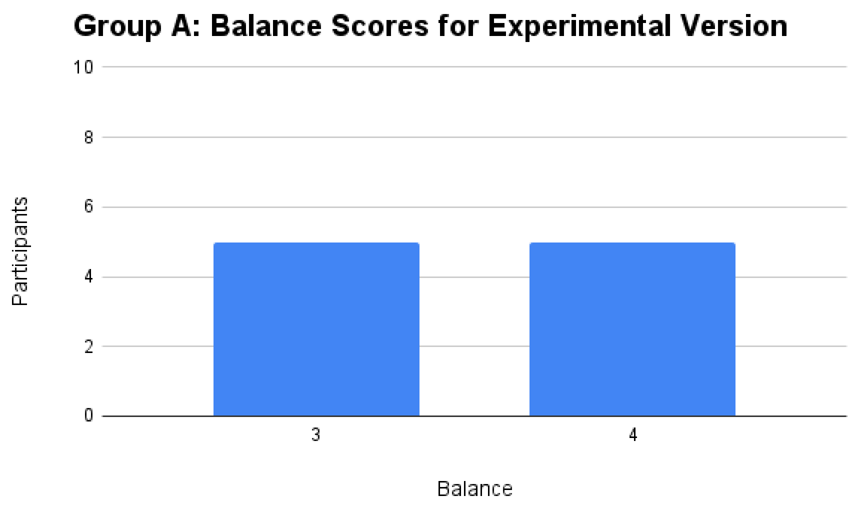
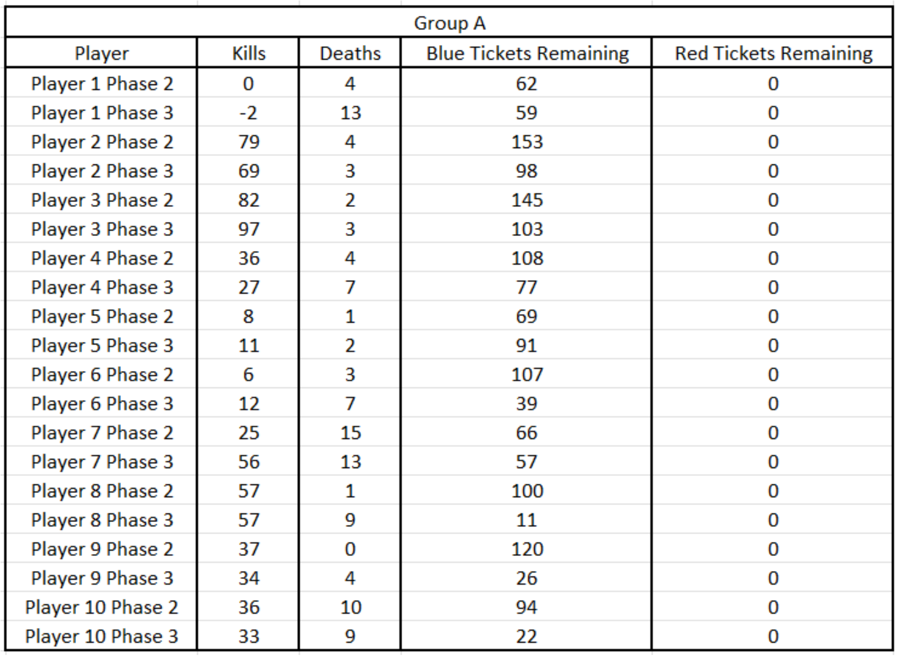


Figure 4.13: *Balance scores for the experimental version from Group A's participants.*

4.2.9 End Game Statistics

From the end game statistics, the kill-death ratio for all participants was derived. It was noted that while 7 participants performed slightly worse, 3 participants performed slightly better, despite still registering an increase in difficulty. Out of the participants that improved, one had an adaptability score of 4, whereas the other two had a score of 3 and 2 respectively. This signals that self-assessment can be an inaccurate way for developers to base their game's ideal difficulty setting. All the participants of this group managed to win both games played. 9 of these 10 participants had less lives remaining when finishing the experimental game, as compared to their lives remaining when finishing the original game. The outlying participant here, through data gathered in Phase 1, is identified as having no playtime weekly, with an adaptability score of 3. It seems that the DDA system could not react as required to provide this participant with a more balanced match.



Group A				
Player	Kills	Deaths	Blue Tickets Remaining	Red Tickets Remaining
Player 1 Phase 2	0	4	62	0
Player 1 Phase 3	-2	13	59	0
Player 2 Phase 2	79	4	153	0
Player 2 Phase 3	69	3	98	0
Player 3 Phase 2	82	2	145	0
Player 3 Phase 3	97	3	103	0
Player 4 Phase 2	36	4	108	0
Player 4 Phase 3	27	7	77	0
Player 5 Phase 2	8	1	69	0
Player 5 Phase 3	11	2	91	0
Player 6 Phase 2	6	3	107	0
Player 6 Phase 3	12	7	39	0
Player 7 Phase 2	25	15	66	0
Player 7 Phase 3	56	13	57	0
Player 8 Phase 2	57	1	100	0
Player 8 Phase 3	57	9	11	0
Player 9 Phase 2	37	0	120	0
Player 9 Phase 3	34	4	26	0
Player 10 Phase 2	36	10	94	0
Player 10 Phase 3	33	9	22	0

Figure 4.14: *End Game statistics for Group A's participants.*

Original Game Average Kill-Death Ratio	17.9
Experimental Game Average Kill-Death Ratio	8.9
Original Game Average Lives Remaining	102.4
Experimental Game Average Lives Remaining	58.3

Table 4.1: *Average scores calculated from the results shown in Figure 4.14.*

4.3 Questionnaire Results for Group B

This group played the experimental game, followed by the original game. The feedback provided from the questionnaire in Phase 1, helps to give a better understanding of the demographics of the group. All comparisons will be compared as carried out during the experiment, first discussing the experimental game's results before moving on to the feedback obtained about the original game.

4.3.1 Play Time

This group was adequately balanced, however, it did not have any participants that spend 20+ hours per week engaged in playing Digital Games. In Figure 4.15, two segments represent 2 participants each, whilst another 2 segments represent 3 participants each.

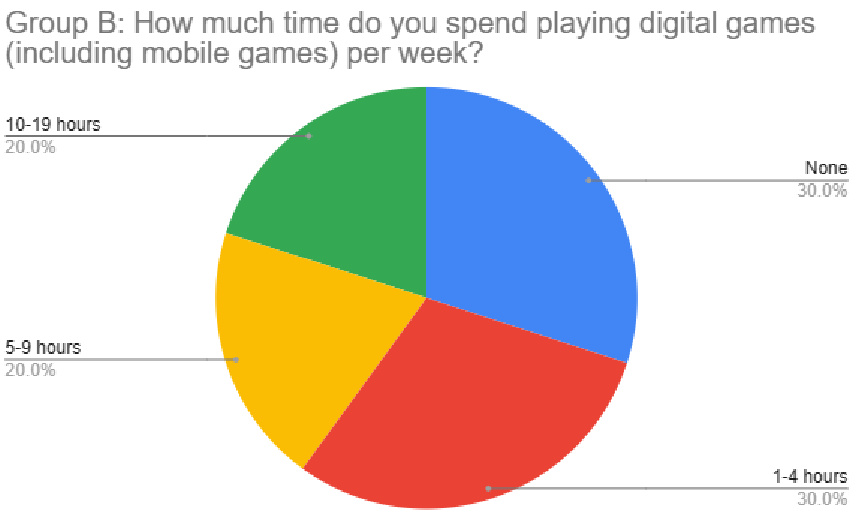


Figure 4.15: *Distribution of Group B's participants' playtime.*

4.3.2 FPS Game Skill Rating

Figure 4.16 shows that half of the participants rank themselves as having the least possible skill in an FPS game. This was ideal, as this group played the experimental version first. This provided insight into how the implemented DDA reacts to novices, without having to worry about the effects of adaptability.

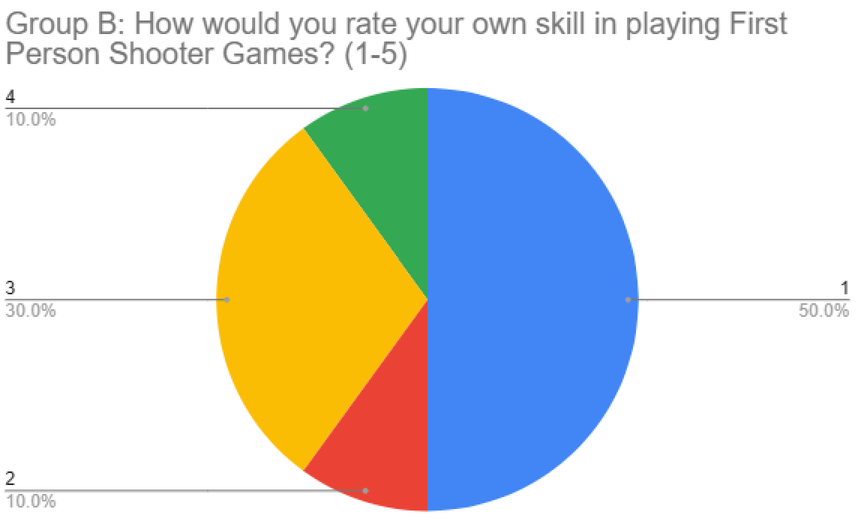


Figure 4.16: *Self-assessments from Group B's participants on their skill level in FPS.*

4.3.3 Adaptability

Figure 4.17 explains the Group B's self-assessment regarding their adaptability. Six participants agreed with having a normal level of adaptability. One participant agreed with being extremely adaptable by selecting the value 5. Another two participants and the remaining participant agreed with the values 1 and 2 respectively. The average adaptability score for Group B was derived to be 2.7 out of 5. Therefore, a noticeable improvement should be visible between the first and second play session.

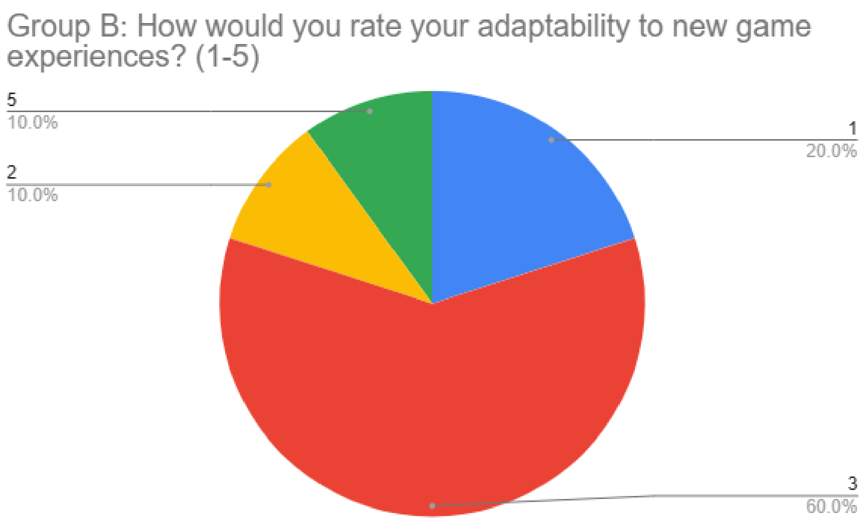


Figure 4.17: *Self-Assessments from Group B's participants on their level of adaptability.*

4.3.4 Game Difficulty Preferences

Figure 4.18 shows a wider distribution than that observed in Group A. Half of these participants prefer a challenging game, whilst the other half either prefer an easy difficulty or an extremely challenging difficulty. This again proves the value of having a well-balanced and challenging difficulty. It also builds on the argument that no single difficulty setting is suitable for all players.

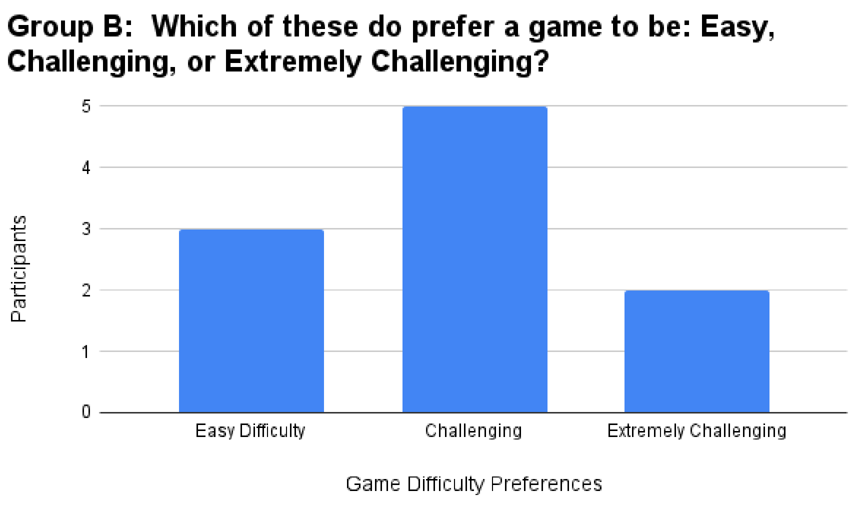


Figure 4.18: *Difficulty preferences for Group B's participants.*

4.3.5 Challenge Scores

Comparing the feedback shown in Figure 4.19 and Figure 4.20, a slight increase in the average difficulty was noted by participants when playing the original version.

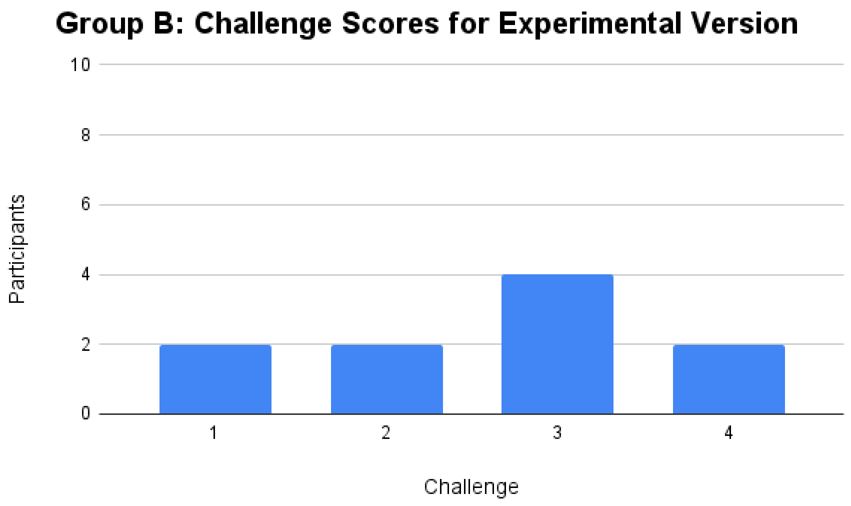


Figure 4.19: *Challenge scores for the experimental version from Group B's participants.*

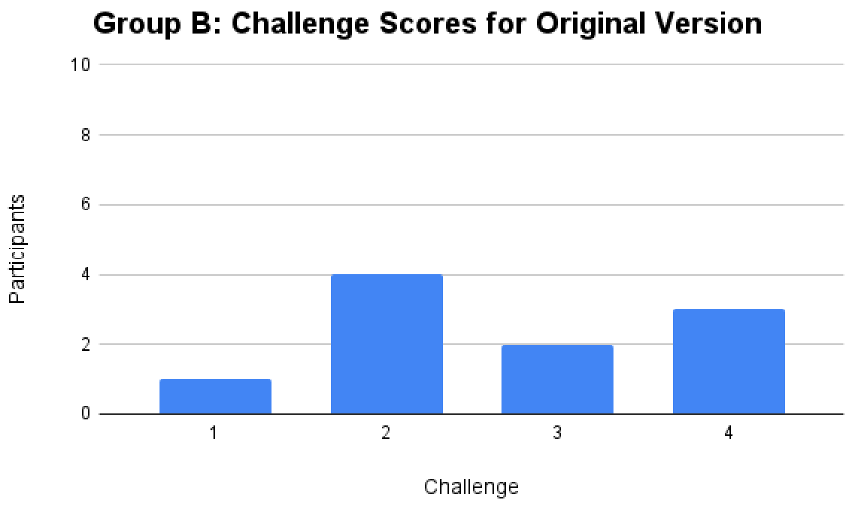


Figure 4.20: *Challenge scores for the original version from Group B's participants.*

4.3.6 Engagement Scores

Figure 4.21 and Figure 4.22 show that a minor increase in engagement was felt by the participants when playing the original game. The experimental version was given an average score of 4.1 out of 5, whilst the original version was given an engagement score of 4.2 out of 5.

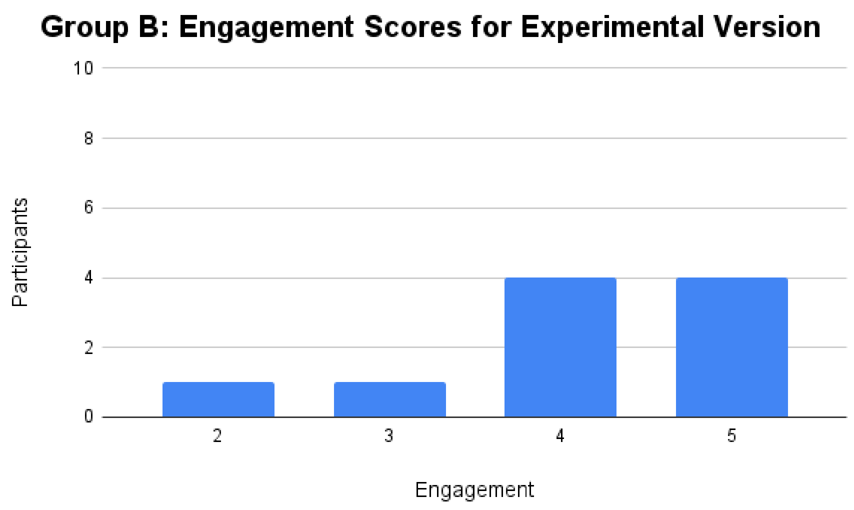


Figure 4.21: *Engagement scores for the experimental version from Group B's participants.*

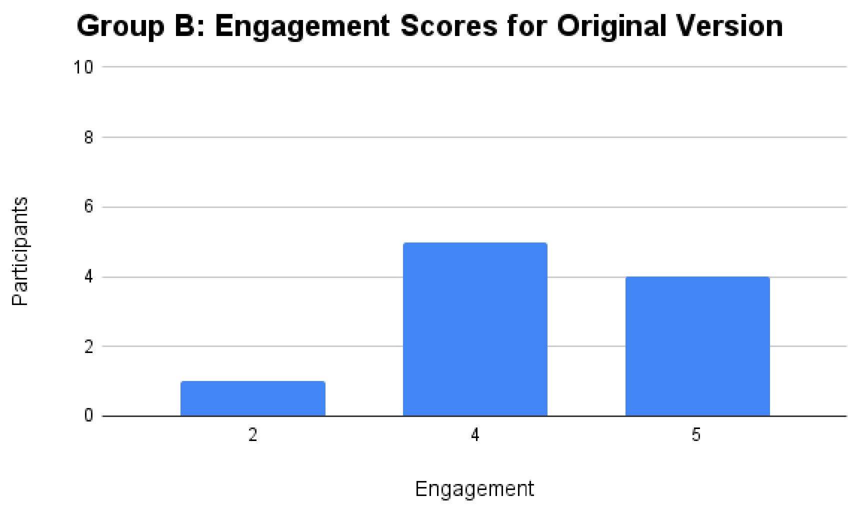


Figure 4.22: *Engagement scores for the original version from Group B's participants.*

4.3.7 Enjoyment Scores

Participants felt no changes in enjoyment between the two versions of the game, as can be seen in Figure 4.23 and Figure 4.24.

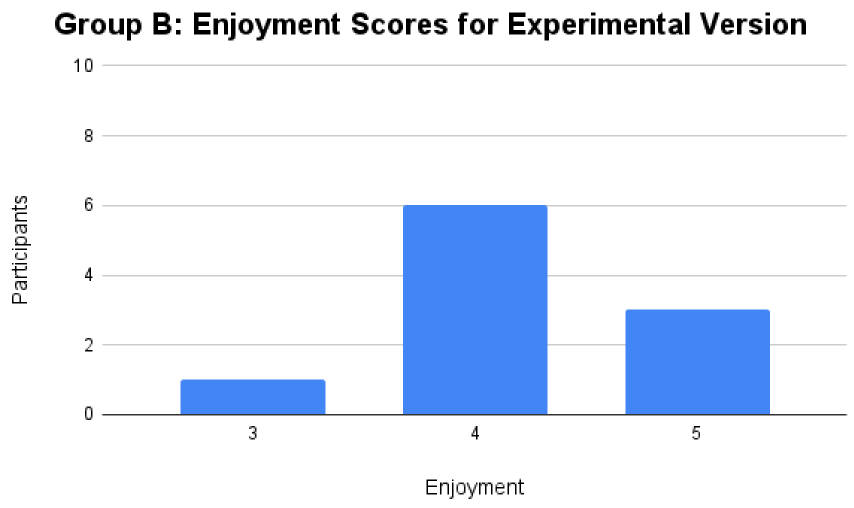


Figure 4.23: *Enjoyment scores for the experimental version from Group B's participants.*

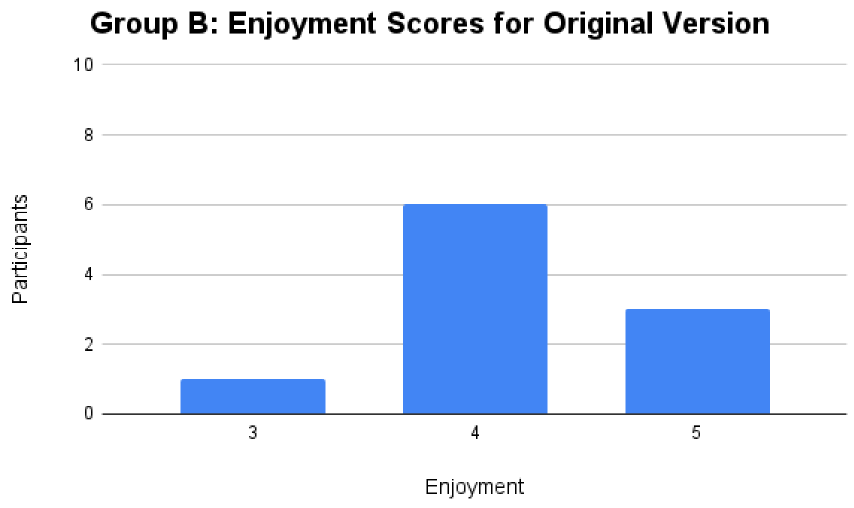


Figure 4.24: *Enjoyment scores for the original version from Group B's participants.*

4.3.8 Balance Scores

As shown in Figure 4.25 and Figure 4.26, participants felt a decrease in team balance when playing the original version. The experimental version scored an average of 3.1 out of 5, whilst the original version scored an average of 2.8 out of 5.

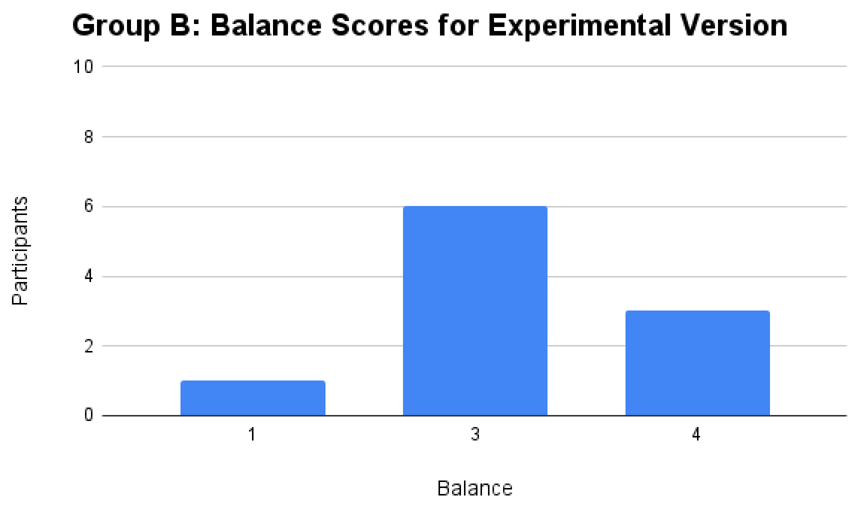


Figure 4.25: *Balance scores for the experimental version from Group B's participants.*

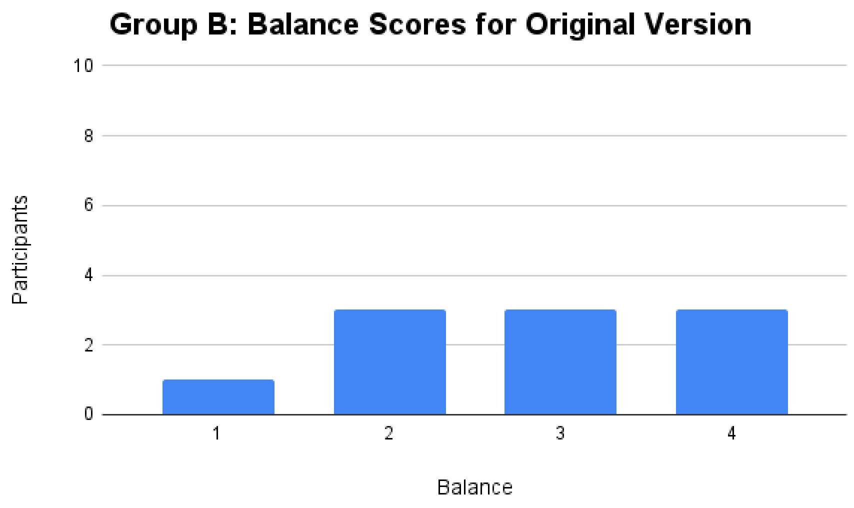
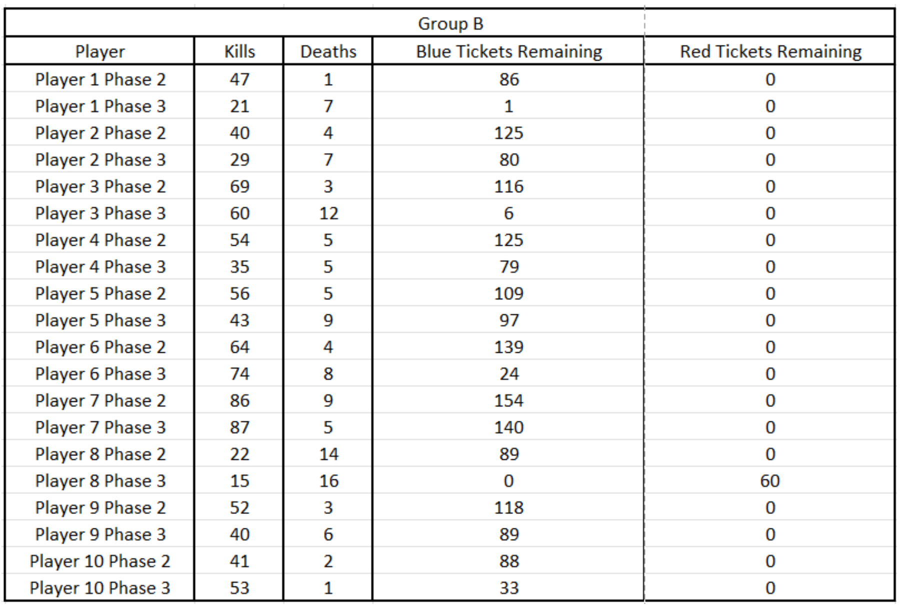


Figure 4.26: *Balance scores for the original version from Group B's participants.*

4.3.9 End Game Statistics

From the end game statistics, the kill-death ratio for all participants was derived. During the experimental game, eight participants performed slightly worse, however, two participants performed slightly better. Considering their improved performance, one of these participants still felt that the experimental game was more challenging whilst the other participant felt it easier than the original version. One of the experimental games played by this group ended in defeat, potentially due to the enemy team having gained the flag advantage. This indicates that in some cases, the DQN model was not adjusting the difficulty as much or as fast as required.



Group B				
Player	Kills	Deaths	Blue Tickets Remaining	Red Tickets Remaining
Player 1 Phase 2	47	1	86	0
Player 1 Phase 3	21	7	1	0
Player 2 Phase 2	40	4	125	0
Player 2 Phase 3	29	7	80	0
Player 3 Phase 2	69	3	116	0
Player 3 Phase 3	60	12	6	0
Player 4 Phase 2	54	5	125	0
Player 4 Phase 3	35	5	79	0
Player 5 Phase 2	56	5	109	0
Player 5 Phase 3	43	9	97	0
Player 6 Phase 2	64	4	139	0
Player 6 Phase 3	74	8	24	0
Player 7 Phase 2	86	9	154	0
Player 7 Phase 3	87	5	140	0
Player 8 Phase 2	22	14	89	0
Player 8 Phase 3	15	16	0	60
Player 9 Phase 2	52	3	118	0
Player 9 Phase 3	40	6	89	0
Player 10 Phase 2	41	2	88	0
Player 10 Phase 3	53	1	33	0

Figure 4.27: *End Game statistics for Group B's participants.*

Original Game Average Kill-Death Ratio	16.7
Experimental Game Average Kill-Death Ratio	11.1
Original Game Average Lives Remaining	114.9
Experimental Game Average Lives Remaining	54.9

Table 4.2: *Average scores calculated from the results shown in Figure 4.27.*

4.4 Interview Results

Before conducting a short interview, the participants were individually briefed on the concept being studied and on the differences between the games they had played. However, they were not told which one of the games played used the implemented DDA system.

4.4.1 DDA System Recognition

- Could you guess which of the matches played had Dynamic Difficulty Adjustment?

When asked to guess which of the games played used the DDA system, only 11 out of 20 participants managed to associate the explained characteristics of DDA to the correct game. Some of the casual to expert players reasoned that the experimental match had adapted to become more challenging in order to match their skills, and therefore, deduced that the harder match of the two played had to be the match that utilised DDA. One of these participants also mentioned that the game became visibly harder as it progressed. Another respondent from Group A, deduced that the second game had this adjustment because although the commands and mechanics were still being discovered, the first game felt easy, whilst the second game, even though it was now familiar, still felt significantly harder. To the contrary, beginners and novices reasoned that the DDA managed the game's difficulty by reducing it to match their skill, and so determined that the easier match was the one that employed DDA. This result, together with data gathered through the last question in both Phase 2 and Phase 3, clearly indicated that "Change Blindness" as described by [21], was implemented well as only one more than half the briefed participants managed to make any association that led to the correct guess.

4.4.2 The Model's Balancing Effectiveness

- Do you feel that the adjustments were suitably balancing the game?

At this point the participants had been informed which of the games used the DDA system. Most of the participants stated that the game felt more balanced than the original version. A participant also mentioned that the final game results were a clear indicator that the experimental game was balanced differently and more accurately. Another respondent remarked that this improved balance is especially suited for beginners being introduced to the game, facilitating player improvement. Having an increase in balance was also reported by one participant as having increased the game's difficulty. Contrastingly, one participant felt that in the original game, the teams were more accurately balanced. Another respondent remarked that while an improvement in balancing was felt, there is room for improving its effectiveness.

4.4.3 Participant Performance

- Do you feel that the adjustments effected your performance?

11 out of 20 participants felt that they played better in the experimental version for a multitude of reasons. Some participants credited this increase in performance to an increase in motivation, either due to goals such as improving the kill-death ratio from their first game, or because of the team's remaining tickets being closer due to DDA. Another group of respondents believe that their performance improved to match and surpass the adjusted enemy team. Other participants improved through utilizing different strategies such as standing out of the line of fire and flanking the enemy. A novice participant credits the improvement to the game adjusting such that it became easier. 6 participants felt no change in performance, while the remaining 3 participants felt that they performed better when playing the original game. One reason for this is that the increased difficulty also elevated some participants frustration, which in some cases, resulted in worse performances and strategies.

4.4.4 Additional Effects

- Did the adjustment effect your enjoyment, frustration, or perception of the game?

Some respondents explained that when the DDA system raised the enemy team's difficulty, a concurrent increase in enjoyment and frustration was felt. These participants explained further that the added frustration is beneficial as it produces and maintains a higher level of engagement, facilitating players to get into the "Flow", as explained by [22].

4.4.5 Examining the Perceived Impact of DDA

- What are your general thoughts on Dynamic Difficulty Adjustment? Do you think it is Beneficial or Detrimental to the game playing experience?

The gathered results present a variety of perspectives regarding DDA and its impact on the game playing experience. Some respondents expressed that DDA is particularly beneficial for new players because it eases the player into the game by adjusting to match his/her skill. It was also remarked that the adjustable nature of DDA is useful because it facilitates learning and growing based on the user's skill evolution. Another highlighted benefit is that DDA could strike and maintain a balance between being frustratingly difficult and excessively easy. Another participant stated that DDA encourages beginners to feel better than they actually are, therefore increasing enjoyment and engagement. An enthusiast of simulation racing stated that they would be highly appreciative if such a system existed for the simulation racing genre. This genre requires a considerable amount of practice before a player can be competitive online. The participant feels that such a system would be a highly beneficial training tool. On the other hand, a few respondents had reservations about DDA. One respondent felt that being unaware of the adjustments, could result in the players feeling less capable than they actually are. Another participant mentioned that part of the fun is noticing improvements. One possible solution for this is the creation of a visible post-match skill meter which takes into consideration the player's statistics throughout the game multiplied by the enemy difficulty during different stages of the game. For training scenarios, training sessions using DDA could be ended by playing a game with a universal difficulty setting, as to enable comparison with oneself and others. On the other hand, a respondent felt that being aware of the adjustment could lead players to lose focus and not play at their best, as the DDA will adjust the game to match their input. Another concern was that such a system restricts the player's freedom. Overall, the majority of participants considered DDA to be beneficial, emphasizing its ability to enhance the gaming experience.

4.4.6 The Emotional Effects of Victories and Defeats when using DDA

The participants that agreed with DDA being beneficial were then presented with a scenario, in which they are aware of DDA being active in a game. They were asked to contemplate how they would feel in the case of being defeated because of an enemy being adjusted to match their skill. There were contrasting view, with some participants feeling slightly cheated, while others felt that such a scenario would only motivate them to improve their skills or tactics. Some respondents shared that it would not be upsetting to lose sporadically, however, if the player is being constantly defeated, then then implementation is detrimental as this will cause a steep increase in frustration. Beginners and novices were asked to contemplate a similar scenario, where they are aware that the game is adjusting to become easier, and they manage to win. Some felt that this is welcome because it enhances engagement, whilst other felt that this reduces the value and pride associated with a win. On further discussion some explained that being aware of DDA would have different effects, depending on the character and the motive for playing the game.

4.4.7 Further Remarks

With regards to using DDA as a tool for player improvement, one participant explained that for improvement to take place, ideally you are faced with a harder challenge than you can consistently handle. In future research, DDA systems that are dependent on the player's desired goal could be created in a way that users can choose different DDA priorities, such as facilitating player improvement, enhancing player enjoyment, or to accurately match the participants skill as attempted in this research. One participant also mentioned that DDA could be a setting which the player can turn on and off as desired. One final suggestion was that in the context of an FPS shooter campaign, levels could start at a fraction of the user's recorded skill and ramp up until the game reaches or slightly exceeds the user's level when faced with a boss level.

4.5 Comparison to Previous Studies

4.5.1 Score Comparisons

The version which utilised the developed DDA system resulted in an increased number of deaths for most of the participants. This highlights an increase in difficulty unlike the results of [20] where the deaths decreased due to a decrease in game difficulty. In both cases, a Comfort policy was followed, however in the case of [20], the game was initially challenging, and so the adjustment was aimed at reducing the difficulty. For this research, the original game was set to its normal difficulty, which was effortlessly defeated by the majority of participants, including some which were being introduced to the FPS genre for the first time. Although, in some cases, the game was adjusted to increase difficulty, it still observed the Comfort Policy. This is confirmed by the number of wins from the total adjusted games played being 19 out of 20. While the results of [20] showed an increase in engagement for experts, this research recorded an equal average score for beginners, while registering a slight decrease in casual and expert players, as seen in Table 4.3 and Table 4.4. The beginners registered a slight increase in enjoyment, whilst the casual/expert players scored the same enjoyment score for both games. The research by [22] registered both an increase and a decrease in frustration depending on different player types. The data obtained through Phase 4 suggests that some participants felt a slight increase in frustration which was described as beneficial. Some felt that this increase helped them to immerse themselves further into the game. This increase could have negative implications in the long-term.

	Beginners Original	Beginners Experiment
Challenge Score	2.73	2.82
Engagement Score	4.27	4.27
Enjoyment Score	4.18	4.27
Balance Score	2.72	3.09

Table 4.3: *Average scores of beginner participants split on self-assessed skill level.*

	Casuals/Experts Original	Casuals/Experts Experiment
Challenge Score	2.44	3.44
Engagement Score	4.11	4
Enjoyment Score	4.22	4.22
Balance Score	2.55	3.55

Table 4.4: *Average scores of casual/expert participants split on self-assessed skill level.*

4.5.2 Change Blindness

The DDA was unnoticed by the majority of the participants. A large number of factors could have facilitated “Change Blindness”. One setting that might have helped, is the number of characters, which was set to being 25 for each team. The chaos and unpredictability that this number of NPCs caused was a smoke screen for the adjustments happening in the background, as the players’ attention was focused elsewhere. Figure 4.28 shows how only 4 participants managed to identify the shift in difficulty as strange when compared to the previously played battle. Most credited their adaptability or accused the NPC teammates of under-performing on the second game played. The implementation by [20] utilised the “Change Blindness” effect such that it was not noticed by any users. In the case of this experiment, 4 participants noted a variance in difficulty. This signifies that further research on implementing DDA through AI must be carried out in a context where the causes and effects of “Change Blindness” are investigated rigorously.

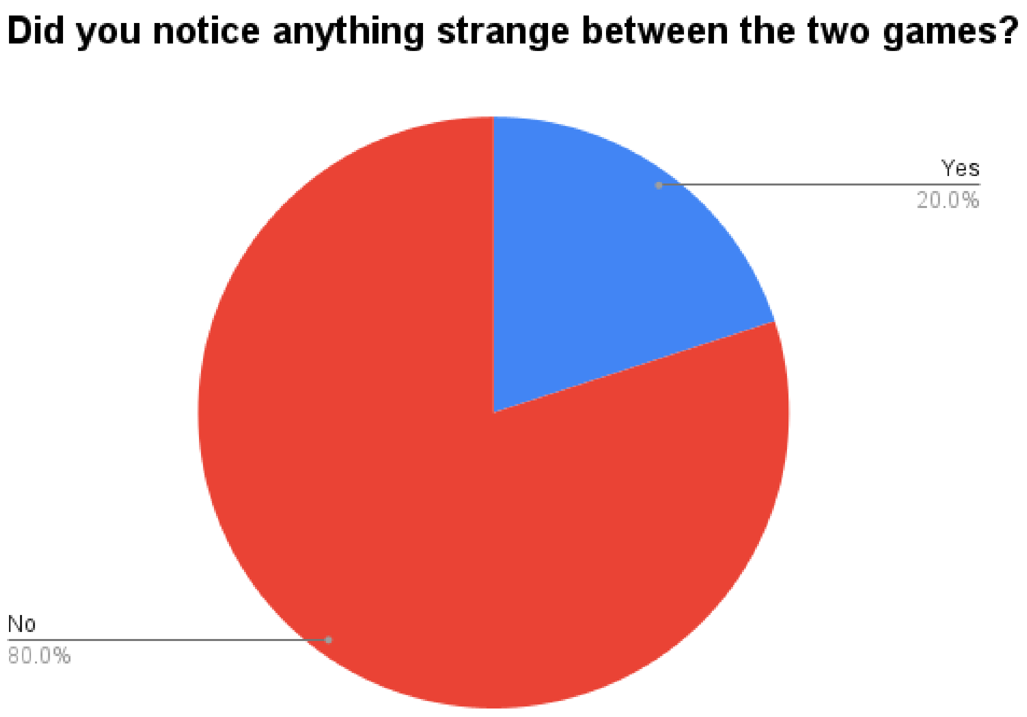


Figure 4.28: *Participants' responses with regards to whether they could identify a difference between the games played.*

Chapter 5: Conclusions and Recommendations

5.1 Research Findings

The study endeavored to address three research questions which relate to the effects of DDA on players' performance, engagement and enjoyment. The implementation also aimed to stay anonymous during adjustments. The results show that for the majority of users, the adjustments were imperceptible. The number of characters per team is likely to have contributed to keeping the DDA system hidden, as the players' attention was focused elsewhere. On average, player engagement was found to have slightly decreased, whilst enjoyment increased slightly. Most of the participants reported an improvement in performance, either due to the elevated challenge which boosted motivation, or through the reduced challenge which gave more space to beginners, in order to effectively grasp the game's mechanics. Some interesting suggestions were made by the participants, such as using DDA for training purposes.

5.2 Limitations

Due to the subjective nature of the research, in which participant views have a considerable effect, it was a lengthy process to develop an experiment which extracts the desired data. Finding relevant connections in the data was also a complex and time consuming task. This research also developed two DQN models, which took a number of days each to train, during which the main device on which the research was being developed, was rendered unusable due to the high processing demand.

5.3 Recommendations for Future Research

The participants mentioned a number of implementations which they believe could have a positive impact on digital games as a whole. One such recommendation is to develop a DDA system which is customisable in its intent and use purpose, as to satisfy different player preferences. Further research must delve deeper into the psychology of players' standpoints, in order to achieve a deeper understanding of the contrast in players' preferences and how this could be tackled effectively. Finally, a similar implementation can be undertaken, with the addition of normalising the observation space to represent a value between 0 and 1. Due to a smaller observation space, the model might train more efficiently and produce better, and more accurate adjustments. Such results could highlight the performance difference that occurs when using a smaller state space in a DQN model.

5.4 Research Conclusion

In conclusion, the chosen methodology effectively addressed the research objectives, combining quantitative surveys and qualitative interviews to deeply evaluate player appreciation towards a Dynamic Difficulty Adjustment implementation that is managed through Artificial Intelligence. The mixed-methods design provided comprehensive insights, ensuring data reliability and validity. This research hopes to have provided a base on which other research may continue to develop and expand the body of knowledge with regards to implementing an objectively satisfying and effective Dynamic Difficulty Adjustment system.

List of References

[1] Skinner Geoff and Walmsley Toby. Artificial intelligence and deep learning in video games a brief review. *International Conference on Computer and Communication Systems (ICCCS)*, 4:404–408, 2019.

[2] Smith Chris, McGuire Brian, Huang Ting, and Yang Gary. The history of artificial intelligence .

[3] Jerry Kaplan. *ARTIFICIAL INTELLIGENCE WHAT EVERYONE NEEDS TO KNOW*. Oxford University Press, 2016.

[4] Andrew Hodges. Alan turing and the turing test. *Parsing the Turing Test*, pages 13–22, Nov 23, 2007.

[5] Mohssen Mohammed, Muhammad Badruddin Khan, and Eihab Bashier Mohammed Bashier. *Machine Learning : Algorithms and Applications*. 1 edition, 2016.

[6] Kotsiantis Sotiris. Supervised machine learning: A review of classification techniques . *Informatica*, 31:249–268, 2007.

[7] Fatemeh Zahedi. An introduction to neural networks and a comparison with artificial intelligence and expert systems. *Interfaces (Providence)*, 21(2):25–38, Mar 1, 1991.

[8] Dayan Peter. Unsupervised learning. *The MIT Encyclopedia of the Cognitive Sciences.*

[9] Jonathan Schaeffer. A gamut of games. *The AI magazine*, 22(3):29–46, September 22, 2001.

[10] Manish Anand Yadav, Yuhui Li, Guangjin Fang, and Bin Shen. Deep q-network based reinforcement learning for distributed dynamic spectrum access. In *2022 IEEE 2nd International Conference on Computer Communication and Artificial Intelligence (CCAI)*, pages 1–6, 2022.

[11] Weiiheng Hu, Jilong Wang, and Yebo Yu. Performance analysis and improvement of deep q network in pommerman agent based on compound training method. In *2021 3rd International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI)*, pages 666–671, 2021.

[12] Chen Wang, Wentao Liu, Qinghao Tian, Shuai Su, and Miao Zhang. An energy-efficient train control approach based on deep q-network methodology. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6, 2020.

[13] Georgios N. Yannakakis and Julian Togelius. *Artificial Intelligence and Games*. Springer International Publishing, Cham, 2018.

[14] P. B. S. Serafim, Y. L. B. Nogueira, C. A. Vidal, and J. B. C. Neto. Towards playing a 3d first-person shooter game using a classification deep neural network architecture. In *- 2017 19th Symposium on Virtual and Augmented Reality (SVR)*, pages 120–126, 2017. ID: 1.

[15] Santiago Ontañón. Case acquisition strategies for case-based reasoning in real-time strategy games. In *Twenty-Fifth International FLAIRS Conference*, 2012.

[16] J. Togelius. Ai researchers, video games are your friends! In *- 2015 7th International Joint Conference on Computational Intelligence (IJCCI)*, volume 1, page 5, 2015. ID: 1.

[17] Glen Robertson and Ian Watson. A review of real-time strategy game ai. *The AI magazine*, 35(4):75–104, Dec 22, 2014.

[18] John E. Laird and Michael van Lent. Human-level ai's killer application: Interactive computer games. *The AI magazine*, 22(2):15–25, Jun 22, 2001.

[19] Chris Fairclough, Michael Fagan, Padraig Cunningham, and Brian Mac Namee. Research directions for ai in computer games, Oct 1, 2001.

[20] Robin Hunicke. The case for dynamic difficulty adjustment in games. *ACM International Conference Proceeding Series; Vol. 265: Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology; 15-17 June 2005*, pages 429–433, Jun 15, 2005.

[21] Daniel J. Simons. Current approaches to change blindness. *Visual Cognition*, 7(1-3):1–15, 2000. doi: 10.1080/135062800394658.

[22] Z. Yang and B. Sun. Hyper-casual endless game based dynamic difficulty adjustment system for players replay ability. In *- 2020 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Comput-*

ing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom), pages 860–866, 2020. ID: 1.

[23] Y. Hao, Suoju He, Junping Wang, Xiao Liu, jiajian Yang, and Wan Huang. Dynamic difficulty adjustment of game ai by mcts for the game pac-man. In *- 2010 Sixth International Conference on Natural Computation*, volume 8, pages 3918–3922, 2010. ID: 1.

[24] J. Y. Wang and Y. R. Tseng. Dynamic difficulty adjustment by fuzzy rules using in a neural network controlled game. In *- 2013 Ninth International Conference on Natural Computation (ICNC)*, pages 277–281, 2013. ID: 1.

Chapter A: Prototype and Resources

All necessary resources are included in a OneDrive linked below. These are available for all necessary utilisation and review.

OneDrive Link:

https://mcastedu-my.sharepoint.com/:f:/g/personal/adrian_mercieca_f31954_mcast_edu_mt/EiREEyk4ooJLq3Wv9-BC7oIBV1PXNpvc6HXmIyQuZNsing