**Literature Review**

***Introduction:***

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].

***The Origins of AI: A Historical Outlook***

***John McCarthy and Alan Turing:***

According to [9], 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.” [2]. 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” [7]. 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 [7].

***Alan Turing’s Philosophy:***

Hodges’s [7] study of Turing’s research identifies a key difference between Turing’s different papers years apart. [7] 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. [7] 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. With this apparent evolution of thought observed in Turing’s releases, Hodges wonders as to when in fact Turing adopted the idea that computable operations would encompass everything normally called “thinking” rather than only “definite method”.

***The Mathematical Objection:***

Hodges [7] 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. Therefore, every algorithm, employed by human or machine is bound to fail occasionally. [7]’s 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”.

***Different Categories of AI Development:***

***Learning Techniques:***

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

***Supervised Learning:***

The authors in [1] [3] 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 [4]. The algorithm of AI that most accurately mimics how the human brain works is the Neural Network. The structure of NN is based on the knowledge we have on how the human brain functions. [6] details the structure of Neural Networks 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. [1] goes 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.

***Unsupervised Learning:***

On the other hand, [3] explains 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 [5].

***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. [15] explains that the agent learns through trial and error to optimize actions with the aim of maximising reward values. [1] explains that this continuous feedback allows it to learn to derive the best possible output.

***Comparison of Learning Techniques:***

[4] 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 all possible actions in an attempt to maximise rewards.

***AI Model Evaluation:***

***The Turing Test:***

Alan Turing is also known for inventing the Turing Test, nicknamed “The Imitation Game”, which is a long-term benchmark for AI Development. The test involves an AI which tries to imitate a human in several tests, with the ultimate goal of being mistaken for a human [7]. [8] explains that the fact that the Turing test is still being discussed and that researchers are still attempting to create software capable of passing it, shows that Alan Turing and the suggested test contributed a powerful and helpful vision to the field of AI. The test requires a computer to not only trick a human into thinking it is intelligent, but also to fool a suspicious human, through conversation. The goal is to be unable to distinguish between man and machine when comparing machine output to actual human output. The communication medium is such that there are no clues beyond what written language may represent. Furthermore, the test does not demand any complicated problem solving, as explained by [8]. At its time of writing, [8] mentions that the Turing Test has not yet been passed. In more recent research, [1] explains that the test was finally beaten in 2014 when the machine took on a persona of a 13-year-old child. The machine convinced a third of the judges by completely avoiding answering the questions using intelligent conversation, resorting instead to the use of sarcasm. The test was technically passed, but some researchers are sceptical on whether this counts as a breakthrough for AI development.

***The New Turing Tests:***

This calls for a new standard when testing for intelligence in machines and [10] outlines 4 different tests which could possibly replace the Turing Test, first of which is the “Winograd Schema Challenge” where a simple but ambiguously worded natural-language question tests a machine’s common sense. It provides the machine with a scenario such as: “The city councilmen refused the demonstrators a permit because they advocated violence.”, after which the machine is requested to answer a question such as: “Who advocated violence?”. [10] explains that these are known as pronoun disambiguation problem and that most people would use “common sense” or “world knowledge” about the provided scenario and its characters to arrive to a conclusion. The second possible candidate is called “Standardized Testing for Machines” and involves the AI going through a standardized, written educational test, similar to what is given to elementary and middle school students. Next, [10] explains another replacement option called the “Physically Embodied Turing Test”. This test entails having an AI physically manipulate real world objects to build a structure after which the AI is then required to explain its efforts. The author recognizes that an AI with this level of competency is well beyond the current state of the art, however, hope that such a test would set a precedent to the importance of integrating the four strands of AI: Perception, Action, Cognition, and Language. The final candidate mentioned is the “I-Athlon”, where an AI is asked to summarize the information from an audio file, condense the contents of a video, and translate natural language amongst other tasks.

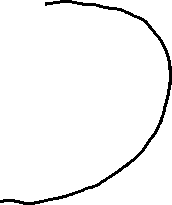
***Video Games:***



***Experience***:



[19] states that video games are designed to provide compelling experiences, through player interaction with a game’s internal systems, such as the inventory and its contents, which has a direct impact on the player’s experience.



***Real-Time Strategy Games***:

The Real-Time Strategy (RTS) game genre consists of some of the most popular and demanding games where the player must control and manage a number of different units and structures with the ultimate goal of developing a powerful enough force that can win against opposing forces [15]. The author goes on to explain that in RTS games, the players start with a very limited number of resources and controllable units which they must coordinate to gather more resources from the game map and use those resources to further develop their force, through building infrastructure or training additional units. This causes a complexity unlike most other game genres where the player must use both Tactical Decision Making (TDM) and Strategic Decision Making (SDM). TDM, also known as micromanagement, is any short-term planning necessary for more immediate circumstances, such as relocating units in order to maximise their effectiveness in a battle. SDM, also known as macromanagement, includes any long-term strategic decisions necessary to build a force, such as which units and structures to produce at a given time in play. Human abilities to abstract, reason, learn, and plan, allow us the freedom of adaptability [17], which experienced players are using to outmanoeuvre even the best AI agents developed for this highly complex game field [15].

***AI and Video Games:***

***The Introduction of AI to Board Games:***

[8] explains 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 to it play Chess, which along with Checkers, [8] describes as the games on which most such research was 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.

***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. [15] explains that unlike simple board games, video games often have real-time limitations that blocks 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.

***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. [9] states 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 [1] believes to have been caused by a lack of motivation by 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]. The authors of [1] go on to state that OpenAI’s model was trained using rapid simulations totalling 180 years of game time daily, pitting it against itself and the occasional human using Reinforcement Learning.

***Human vs AI***:

Today, some of the best DotA 2 players have been beaten by machines equipped with deep learning algorithms [1]. The authors of [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.

***Progress of Research on AI in Video Games***:

In [12], released in the early 2000’s, the author highlights 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. [12] also noted that graphical improvement was slowing down, and that no huge leaps in graphics as seen in the release of Doom (1993) would happen again. This means that a shift in priority could occur, with AI being one of the top contenders. [9] and [12] 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, [1] outlines that powerful GPUs are now handling the load presented by ever increasing levels of graphics, which frees up the CPU resources to focus on utilizing more powerful AI systems [12]. [12] 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 system in games. [15] 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 [18].

***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 [13]. [14] outlines 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 tests. The author goes 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 is used to give game designers immediate feedback and suggestions, thus scaffolding the game design process. [16] also highlights 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 [11].

***Dynamic Difficulty Adjustment:***

***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 [19]. 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 [20].

***Precautions for DDA:***

[19] 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 when such adjustment disrupts u degrades 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 [19]. Taking correct measurements where 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.

***Implementations:***

***Implementation 1:***

[19] 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. [19] 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. [19] 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 pre-defined 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. [19] 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.

***Implementation 2:***

[21] explains 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. [21] argues 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 difficulty 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 [21] 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 hypothesises that tackling these elevated, negative emotions through DDA could help keep the player in the “flow”, allowing the player to play 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 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.

***Experiments:***

[19] 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 [19]. Adversely, [21] had participants first try one version of the game, and then 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.

***Results:***

The experiment conducted by [19] 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 [21] 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. [21] shows confidence that with further research, this too can be improved. Whereas the study by [19] implemented DDA through a change in mechanics, [21] added DDA by modifying the game's dynamics, therefore changing its structure. Both approaches are correct, however, one leaves a larger footprint on the game’s aesthetics. [19]’s approach maintains the game’s format, however risks becoming a nuisance for eagle eyed players which notice the adjustment. [21]’s approach does not attempt to stay anonymous and therefore, does not risk the user feeling cheated by the game. Instead, the is implemented such that it is an integral part of the game. A disadvantage to this is that it could lessen the enjoyment of the game and heavily change the experience provided by the original.

[1]

S. Geoff and W. Toby, “Artificial Intelligence and Deep Learning in Video Games A Brief Review,” vol. 4, pp. 404–408, 2019, doi: 10.1109/ICCCS46626.2019.

[2]

J. Kaplan, ARTIFICIAL INTELLIGENCE WHAT EVERYONE NEEDS TO KNOW. Oxford University Press, 2016, p. 30.

[3]

M. Mohammed, M. B. Khan, and E. B. M. Bashier, Machine Learning : Algorithms and Applications , 1st ed. 2016, pp. 20–21.

[4]

K. Sotiris, “Supervised Machine Learning: A Review of Classification Techniques ,” vol. 31, pp. 249–268, 2007.

[5]

D. Peter, “Unsupervised Learning.”

[6]

F. Zahedi, “An Introduction to Neural Networks and a Comparison with Artificial Intelligence and Expert Systems,” vol. 21, no. 2, pp. 25–38, Mar. 1991, doi: 10.1287/inte.21.2.25.

[7]

A. Hodges, “Alan Turing and the Turing Test,” pp. 13–22, Nov. 2007, doi: 10.1007/978-1-4020-6710-5\_2.

[8]

G. N. Yannakakis and J. Togelius, Artificial Intelligence and Games. Cham: Springer International Publishing, 2018, pp. 8–10.

[9]

S. Chris, M. Brian, H. Ting, and Y. Gary, “The History of Artificial Intelligence .” https://courses.cs.washington.edu/courses/csep590/06au/projects/history-ai.pdf#:~:text=The%20term%20artificial%20intelligence%20was%20first%20coined%20by,machines%20can%20truly%20think%20began%20much%20before%20that. (accessed 2023).

[10]

J. Pavlus, “THE NEW TURING TESTS,” vol. 316, no. 3, pp. 61–62, Feb. 2017.

[11]

J. E. Laird and M. van Lent, “Human-Level AI’s Killer Application: Interactive Computer Games,” vol. 22, no. 2, pp. 15–25, Jun. 2001, doi: https://doi.org/10.1609/aimag.v22i2.1558.

[12]

C. Fairclough, M. Fagan, P. Cunningham, and B. Mac Namee, “Research Directions for AI in Computer Games.” Trinity College Dublin, Department of Computer Science, Oct. 01, 2001, [Online]. Available: https://explore.openaire.eu/search/publication?articleId=od\_\_\_\_\_\_\_883::f6ae40c2baa6afbd37ede61c414d2f44.

[13]

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,” 2017, pp. 120–126, doi: 10.1109/SVR.2017.24.

[14]

J. Togelius, “AI researchers, Video Games are your friends!,” 2015, vol. 1, p. 5.

[15]

G. Robertson and I. Watson, “A Review of Real-Time Strategy Game AI,” vol. 35, no. 4, pp. 75–104, Dec. 2014, doi: 10.1609/aimag.v35i4.2478.

[16]

J. Schaeffer, “A Gamut of Games,” vol. 22, no. 3, pp. 29–46, Sep. 2001, [Online]. Available: <https://search.proquest.com/docview/208141813>.

[17]

M. Buro, “Call for AI research in RTS games,” 2004, pp. 139–142.

[18]

S. Ontañón, “Case acquisition strategies for case-based reasoning in real-time strategy games,” presented at the Twenty-Fifth International FLAIRS Conference, 2012.

[19]

R. Hunicke, “The Case for Dynamic Difficulty Adjustment in Games,” pp. 429–433, Jun. 2005, doi: 10.1145/1178477.1178573.

[20]

D. J. Simons, “Current Approaches to Change Blindness,” vol. 7, no. 1–3, pp. 1–15, 2000, doi: 10.1080/135062800394658.

[21]

Z. Yang and B. Sun, “Hyper-Casual Endless Game Based Dynamic Difficulty Adjustment System For Players Replay Ability,” 2020, pp. 860–866, doi: 10.1109/ISPA-BDCloud-SocialCom-SustainCom51426.2020.00133.

**Research Questions**

* Does the adjustment affect player performance?
* Do players notice when adjustments occur?
* Does adjustment significantly affect the player’s enjoyment, frustration, or perception of game difficulty?

**Research Methodology**

**DDA AI Model - Implementation and Evaluation:**

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. Similarly to [19], a Comfort Policy was chosen for this research, where the DDA aims to keep the player relatively safe while keeping the game reasonably challenging. 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. While evaluating the game as an option, It was immediately noticed that the game is instantly challenging and that newcomers are likely to initially find it difficult and frustrating. Therefore this was seen as a good scenario on which to apply DDA, in an attempt to elevate the player experience.

‘Ravenfield’ was developed by Johan Hassel using the Unity Game Engine. Although not great in number when compared to other, more mainstream FPS games, the game’s community is highly active and 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, which involves having both teams start with 300 unit lives (tickets) and 1 flag. A number of additional flags spread throughout the map can individually be conquered by having team members spend a set amount of time in a flag’s vicinity. The team with the most flags captured benefits from an advantage, as 1 ticket is bled from the opposite team each` second, for as long as this flag advantage is maintained, or until it a duration threshold is reached. The game features a small catalogue of weapons and tools from which the player can create a loadout (inventory). NPC’s 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 missiles and present a high power advantage, and other vehicles such as jeeps which offer utility in the larger maps available in the game. For this research, vehicles were excluded as to contain the complexity, both in scale and power.

**Game Changes For Training:**The game was adapted 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 NPC’s with the best set attributes, and with skill level 5 being the worst, therefore equipping the friendly NPC’s with the worst set attributes. [Explain Further on the different levels]. This added 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 level and circumstances. Some testing functionality was added to make sure that skill levels were indeed changing on each restart of the match. In trying to keep the game as simple as possible, a new moderate scale 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 resulted in 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. 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, time since start of match, and current enemy skill level, as set by the AI Model. The time data was used for monitoring purposes, whereas the rest of the 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 later on retrieve important observation data. The ‘GameManager’ class was adapted to retrieve the action selected by the AI Model and save this into a static variable. The 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 presets in this case ranged from 1 to 11, with skill level 1 being the best, therefore equipping the hostile NPC’s with the best set attributes, and with skill level 11 being the worst, therefore equipping the hostile NPC’s with the worst set attributes. This variation between the blue and red team in skill level presets was implemented purposefully so that the Model has flexibility to overcome and adapt to situations as required.

**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, whereas a Python package PyAutoGui allowed for automating button presses and clicks required to restart a ‘Ravenfield’ battle. Finally an OpenAI Gym environment was developed, which coordinated the Model training cycle. 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 organised and clear for debugging. Firstly, the automation process was created using PyAutoGui. ‘Restart’ and ‘Deploy’ button presets were created, which stored four attributes: start of button from left, start of button from top, width, and height. These attributes were calculated through the use of a ruler, which measured the width and height of the on-screen image, which together with the screen resolution were used to find different points of the screen through the formula: (measurement of full screen / resolution) \* on screen measurement related to the button

Example, to find the start of the restart button from the left:

(40cm / 1920p) \* 16cm

Similarly, an example of finding the size of the button’s height:

(25cm/1080p) \* 2cm

Two functions were developed to handle the restart process from start to finish. The first process started by maximising the ‘Ravenfield’ window, and is followed by a small interval [check/specify seconds]. The first process started by autonomously inputting the ‘escape’ button, at which point an in game menu popped up. The mouse was then guided to the hard coded location where the restart button is located, and after a 1 second interval, an autonomous left click started the in-game restart process. Finally, the first function waits 5 seconds before concluded The second function starts by guiding the mouse onto the deploy button [confirm button name], and after a 1 second interval, autonomously executed a left click which initialised a new ‘Battle’. Utilising the previously set buttons, guiding was achieved by using the ‘pyautogui.moveTo(x, y)’ function, where x and y where replaced by a formula that finds the middle point of the button’s x and y axis respectively. Example: pyautogui.moveTo( RESTART\_BUTTON[0] + RESTART\_BUTTON[2]/2, RESTART\_BUTTON[1] + RESTART\_BUTTON[3]/2 )

Following the development of these functions, some testing code was added to the notebook, which was helpful in finalising the button presets, through trial and error. Installation and importation commands were then used to install the ideal ‘stablebaselines3’ package and OpenAI’s ‘gym’ package. Code was added to demo the data reading required from ‘GameState.csv’, which is where the observation data is logged by ‘Ravenfield’.

A gym environment was then built with five necessary functions. Firstly, the function ’\_init\_’ handled 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 enemy difficulty, and a value between 0 and 2 representing peaks. In the initialising function, other variables which played important roles within the other functions, were declared and initialised by a default value. Another function was named ‘get\_observation’ and this handled data retrieval and assignment. Firstly, an infinite loop encompassed everything, inside which a try/catch block is utilised inside which all reading and assigning attempts took place. On a successful attempt, the loop broke and the function moved to return the acquired values. Unit balance was retrieved by subtracting the red tickets remaining from the blue tickets remaining, where a negative value means that the blue team were disadvantaged and a positive value meaning the contrary. The current difficulty was retrieved and subtracted by 1 in order to be refactored to the range accepted by the observation space. The peak was set to 0 when the current difficulty was at its hardest (0) and set to 2 when the current difficulty was set to its lowest (10). Otherwise the peak was set to 1. The flag balance was calculated by first adding the difference in currently owned flags to the total flag balance, which was then divided by the total steps taken since the battle started. An other variable signalled whether or not the match should be restarted, depending on whether the teams still have units remaining or not. Finally this function returned a Dictionary with the needed values to satisfy the environment’s observation space. 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 was rewarded based on its performance as seen through the retrieved data. The step counter used to calculate the flag balance in incremented and the episode length is which prompts an environment reset one it’s value is 0, is decremented. This was followed by an infinite loop that contained a try/catch block which attempted to save the action to ‘AI\_Actions.csv’. On successful saving, the loop is and the function returns the observation, the gained reward and a variable signalling whether the episode is done or not. The environment also required a function to facilitate it’s reset. This function started off with a contingency procedure that maximised the ‘Ravenfield’ window and then waited 2 seconds, therefore making sure that any following button presses were placed within ‘Ravenfield’. At this point, the reset function called the restart and deploy functions declared at the start of the notebook, which carried out a match restart within ‘Ravenfield’. The episode length was then given a value, which depended on how many episodes had taken place beforehand. The episode counter was then incremented and all other variables were assigned their default values. The function then waited 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’ was emptied and an observation is returned using the ‘get\_observation’. The final function included was the empty render function, which was necessarily declared for the environment to be valid and to fully the gym interface.

The gym environment was then subject to testing through manually checking values being stored in temporary text files and subject to validity checking using the ‘check\_env’ function imported from ‘stablebaselines3’.

**Dynamic Difficulty Adjustment Models:**   
The AI Model was created using the “stablebaselines3” library available for Python. This library facilitates the creation of AI models with varying settings. 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]. An observation is the data which the Model associates with an action during training. Initially an observation included: the blue team’s remaining tickets, the red team’s remaining tickets, flag balance and current set difficulty. Later in the research, another 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 after every 1000 steps. This allowed for detailed comparisons between models in terms of effectiveness and improvement rate. Code was added so that the models could be loaded and tested, by having the loaded model play and return the total rewards, which can be compared to other models’ results. The initial Model was trained for 265000 steps and took 5 days of non stop training. The second Model war trained for 200000 steps an took [specify later].

Just as [19] 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. [Explain further on Formulas used]. The second 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 second Model with the aim of encouraging recovery and reward maximisation. One of these conditions required that the unit balance be 10 or less, at which case the Model was rewarded with a value of 3. Similarly, a minimal reward of value 1 was given when the unit balance was 15 or less.

The first model was loaded and tested and showed promising results, however after a more in-depth recheck it was noticed that the flag balance for this model was incorrectly implemented, as instead of dividing the total flag balance by the number of steps taken, it was divided by the seconds passed since the start of the game. After the issue was fixed, the previously mentioned changes were made to the environment’s rewarding system and a second model was trained.

**Gauging User Feedback:**   
A four Phase experiment was planned and conducted, during which valuable data was collected. 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. A group of [specify later] people were invited and accepted to take part in the experiment. As done by [21], the participants were divided into two groups, which differentiated only in the order of the Phases carried out. The first group carried out Phase 2 followed by Phase 3, whereas the second group 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.

A number of questions were prepared for each individual Phase. These questions were the main source for all the primary data collected.

Phase 1 used questions to gauge the participants’ capabilities through self-assessment. The questions were:

1. How often do you play digital games? (1-5)
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 three 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 first 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. The second question provided an estimation of the participant’s skill in similar games, which could later be compared to data logged during 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.

The questions for Phases 2 and 3 were identical, as these gauged opinions related to the gameplay experienced, within the respective Phase. The questions were:

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?

Similarly to Phase 1, 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 complete. 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 [20], was present during play, such that changes in difficulty were not identified.

Phase 4 questions were used to directly gauge the participant’s appreciation towards Dynamic Difficulty Adjustment systems such as the one experienced during Phase 3. The questions were:

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?

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.

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 2:

The participants were equipped with an Apple Watch Series 6 which was used to gather heart rate data which could then be related to levels of engagement. Participants were then 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:

This Phase worked exactly like Phase 2 with a single and important change. During this Phase, the participants played a match using the edited version of the game ‘Ravenfield’ which utilised Dynamic Difficulty Adjustment, as managed through the AI Model. After the end of the match, evaluation was carried out as done in Phase 2.

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.

For all stages of this research, an ASUS ROG STRIX G15 was used with  
the following specifications:  
• Central Processing Unit: Intel I7-11850H  
• Memory / RAM: 16GB  
• Graphical Processing Unit: Nvidia RTX 2060 (Laptop)

**Results**

The experiment conducted by [19] resulted in having a significant decrease in player deaths, with post play evaluations revealing no differences in enjoyment for novice players, whereas expert player 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.