How is Player Enjoyment and Perceived Challenge Affected When Implementing Dynamic Difficulty into a Top-Down Shooter?

Adrian Carter - 1703170

***Abstract*—With the rapid growth of the video games industry and its popularity, players are more varied and diverse than ever before. With this diversity comes a plethora of problems for game developers. One such problem is the effectiveness of traditional static difficulty settings. Static difficulty settings have been standard in most mainstream commercial games. A game with a set difficulty setting such as this will be too hard for some players and too easy for others. However, more and more research is being conducted with Dynamic Difficulty Adjustment (DDA) systems that can adjust difficulty settings during runtime according to the current player's ability. This study aims to test the effect on perceived player experience of a dynamic difficulty system when compared to a static difficulty system. Specifically within the context of a singleplayer, top down shooter level with an average play time of under 3 minutes. Effectiveness is measured based on player experience feedback using questionnaires and telemetry data during gameplay. The results of the experiment have shown that there is a difference in player perceptions between the two difficulty systems. Player enjoyment was shown to be higher when playing the dynamic version of the game, whereas player challenge perception was higher when playing the static version. However, player performance was much more varied in the static version. Despite this, players were unable to reliably identify which level used dynamic difficulty. This seems to indicate that dynamic difficulty systems are more favoured by players and allow players to perform better, at least in the context of the game created for this experiment.**

I. Introduction

THIS research project investigates the effects Dynamic Difficulty Adjustment (DDA) can have on Player Experience, specifically the effect on Player Enjoyment and Challenge perception. With a more diverse variety of people playing video games than ever before [1] more care has to be taken when designing difficulty in video games. However, with the advancements of computational systems, it is becoming easier for developers to manage and adjust a wide variety of parameters in video games, even while the game is running. This has led to the expansion of DDA in recent years, enabling the designer to have more control over different difficulty factors. To adjust the difficulty, many heuristics (techniques to quickly problem solve) can be considered. Heuristics ranging from affective feedback [2] to player performance [3]. These heuristics are discussed in this paper in order to select appropriate methods that might be utilized in the creation of a bespoke dynamic difficulty system. This paper hypothesises that if one were to create two difficulty settings for a 2D top down shooter, one static and the other dynamic, the dynamic setting will be preferred by players. The paper also hypothesizes that the dynamic setting will have less variation in challenge perception and performance for players, since the dynamic setting should adapt better to players of varying skill levels. Section II of the paper will look at some of the theories and concepts that are relevant to the research question. The most prominent aspects that need to be considered when designing a dynamic difficulty system include player experience, flow theory, and player modelling. Section III discusses some of the most notable implementations of dynamic difficulty and why some of those techniques would be useful in designing a dynamic difficulty system for a 2D top down shooter with short levels. Section IV discusses the hypotheses used in the experiment in order to judge player experiences and performance between the two difficulty systems. Section V will discuss the methodology by which this paper intends to answer the proposed research question. This includes a discussion of how the experiment was carried out, what metrics have been used to test the hypotheses and what considerations were made to ensure accurate and reliable results. Section VI explains the design of the Artefact created in order to carry out the experiment. This includes a description of the mechanics in the game and how they were created, a walkthrough of the tutorial and an explanation of various game features. Section VII walks through the development of the Artefact as well as a detailed look at how testing was carried out to verify and validate the two difficulty settings. This testing included the creation on an Artificially Intelligent Player Agent used to simulate playtesters. Section VIII analyses the data gathered during the experiment with regards to the hypotheses. Section IX discusses the results from the experiment in more detail.

II. Related Work

A. Player Experience

Player experience in games is often described using terms such as flow, immersion, engagement, fun, and presence [4] [5] [6]. Measuring this is no easy task due to the subjective nature of player experience. Although tools can be used to objectively measure some quantitative aspects of the player experience such as biosensors to measure heart rates or electrodermal activity (measure of sweat perspiration)[7] [8] [2], they still require evaluation based on the subjective experiences of the individual. When measuring how challenged a player is when playing a game, quantitative statistics can be measured such as the number of deaths and other player performance statistics. However, when measuring the perception of challenge

a player experiences or player enjoyment we would have to find alternative methods. Questionnaires can be particularly useful in measuring the subjective experiences of players in a quantifiable manner, due to the fact they include a number of standardised tests that can be aligned with what the researcher intends to measure about said experience [9]. This leads to the question of how developers can create a good player experience.

To ensure that player experience is enjoyable, the idea of Flow must be considered. Cziksentmihalyi [5] [10] uses the term 'Optimal Experience' to describe the moments in which a person feels a deep sense of enjoyment or exhilaration and states that these moments are, more often than not, due to events in which that person's body or mind is stretched to its limits in an effort to achieve something difficult or challenging. The experience of Flow, or 'Optimal Experience', in games is the experience gamers describe when immersed in video games, becoming entranced and totally focussed on their actions and losing track of time and other worries. Whether a player enjoys a video game is largely dependent on whether that game provides a Flow experience. In Flow theory, the idea of the 'Flow Zone' (see figure 1) is the notion that in order to maintain a players Flow experience, the game must ideally balance challenge and the ability of the player to overcome that challenge. If the game becomes too challenging, the player becomes disheartened and anxious [11]. On the other hand, if the game becomes too easy, the player grows bored and disinterested [12]. The art of good game design, is therefore in the ability of the developer to balance challenge and player ability or skill for the duration of a game experience [13]. This becomes more difficult the larger the game grows and also the larger the potential audience becomes. With a larger potential audience, the fact that each person experiences activities in different ways becomes problematic for game designers. Every player is different, with varying skill levels and expects a different level of challenge in order to experience Flow [5]. The most widely used solution for this problem in video games has been the implementation of various difficulty modes (Easy, Medium, Hard etc) [14]. One of the problems with this is the lack of detail the player is given when choosing a difficulty level. How easy is Easy? How hard is Hard? Without playing through the game, the player won't really be able to know which mode is best suited for their skills, since difficulty ratings are generally subjective [15]. This is where the notion of DDA becomes very useful to developers.

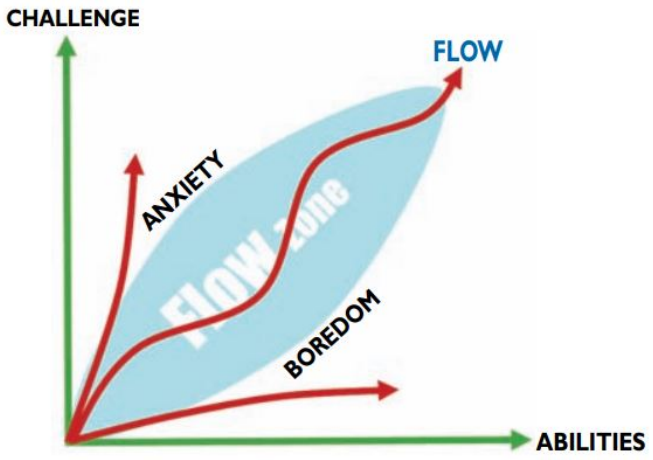

Fig. 1. Chen, Flow in Games [16].

B. Player Experience Modelling

While there has been much research on what constitutes fun in game design and how player engagement lends itself to other aspects of player experience [17] [13], much of this research is qualitative in nature and serves mainly to aid in game design conceptually. Less research exists in the field regarding quantitative measurement of player experience and player modelling. The primary objective of player experience and player modelling research, is to describe how players interact with and experience a game.

Player Experience Modelling, in the context of this paper, refers to the modelling of complex and dynamic aspects of a player interacting with the world during gameplay. Specifically the study and use of AI techniques to create computational models of player experience. This is not to be confused with player profiling, which deals with the types of players you are designing the game for [18].

Yannakakis et al [19] argue that one of the main aims of studying players in games is to gain an understanding of players' cognitive, affective and behavioural patterns. This is strongly related to the field of Human Computer Interaction [20] [21] and usability.

The simplest way of generating a model of player experience is to ask the players themselves and build a model based on the collected data. Yannakakis et al [22] describe this as Subjective Player Experience Modelling. Using short questionnaires like this seems like an effective method to model player experience during the experiment. This study [23] used gameplay interactions to construct a computational model of player experience using statistical features such as how often and when the player jumped, ran, died etc. Whilst it is out of the scope of this paper to collect data such as this during development in order to adapt the dynamic algorithm design, this data collection method would be useful in measuring the effectiveness of said design once complete.

The creators of Polymorph [24] used methods from machine learning and procedural content generation to create a 2D platformer that generates upcoming parts of the level based on the current skill of the player. In order to create the system, a model of difficulty was required. Instead of modelling difficulty based on subjective game design ideas, the creators designed a mass data collection tool to gather information about level segments directly from players. Each player would play a short segment of the level and then give feedback by rating the perceived difficulty on a scale from 1 to 6. The tool would also collect telemetry data on the players behaviour while playing the level segment which could be applied to a machine learning algorithm in order to model player skill. This collection of player behavioural data is something worth considering when testing the effectiveness of the dynamic and

static difficulty settings.

III. Dynamic Difficulty Adjustments

The challenge of designing a difficulty curve for players of varying skill level is a big problem even experienced designers with strong domain knowledge have difficulty solving. In contrast to static difficulty, DDA attempts to address this issue.

Currently, research in the field of DDA is in a state that does not allow for particularly complex systems, and as such, cannot be used as a framework for DDA systems in other games. Although some research is being done to establish DDA frameworks [25], they generally cover only certain basic elements of gameplay, usually within 2D games [26] and more complex variants need to be implemented in bespoke ways, depending on the genre and game elements that need to be adjusted. The ways in which these methods adjust difficulty may differ, but they all exhibit a common theme in prediction and intervention.

A. Dynamic Scripting

Dynamic scripting (DS) is an online learning approach that maximises an AI players chance of winning a game by determining what tactics will lead to the solution, when presented with a certain problem and domain [27]. Using dynamic scripting, the AI player adapts its tactics based on the opponents' tactics using domain-specific knowledge and several rulebases that categorize various opponent types. Each script rule is assigned a weighting value that is adjusted based on the success or failure rates of said rule (see *figure 21*).

The AI learns and adapts progressively using this technique. The AI will update the weight value of the current rules being used during a confrontation with the player, based on the outcome of said confrontation. If the resulting outcome is beneficial to the AI, the rules that lead to that outcome will have their weight values increased, whereas the rules that were counter to that outcome will have their weight values decreased. If the outcome is detrimental to the AI then the inverse occurs. Remaining rules adjust accordingly to keep the sum of all weighted values in the rulebase constant. Eventually, the AI would become too difficult for human players to compete with and so would need to be balanced properly [28]. In order to balance the opponents ability with the players, there are three common ways to modify dynamic scripting:

1) High-fitness penalizing: Increase or decrease weight values proportional to a fitness value. Reward suboptimal behaviour and penalize superior behaviour to essentially make the AI less efficient.
2) Weight clipping: Uses a maximum weight value to limit the rules that are chosen from the rulebase. The higher the maximum value the greater the disparity between the most optimal and least optimal rules. Lowering the maximum weight value during gameplay would make less optimal rules more likely to be chosen, thereby decreasing the effective difficulty of the opponent, and vice versa.
3) Weight culling: As with weight clipping, uses a maximum weight value limiter. However, the weight value of rules are allowed to extend above the maximum value. When a rule has a weight above maximum, the AI is unable to select said rule, thus automatically balancing the AI strength in proportion to the player.

The commercial viability of this Adaptive AI has been dismissed by some developers in the past [29] [30]. The reason for this is the inherent difficulty in creating game AI with online learning capabilities without the fear that the AI will learn inferior behaviour. This is a large concern, particularly to publishers, in the games industry. Simple games with less computational risk have been seen to work [31] and research has been done to effectively implement Adaptive AI using DS into existing commercial games [32]. The following sections describe DDA in use in commercial games today:

1) Resident Evil 4: Resident Evil 4 [33] was created with a hidden grading system that judged the players performance in the background unbeknownst to the player. The grading system is used to classify each player's ability and adapt the game world accordingly. The system grades the player from 1 to 10 with higher grades leading to a higher challenge. If the system detects that the player is playing well by, for example, measuring a high headshot rate or a high survival rate, the player is given a high grading which means the system provides more enemy spawns, more powerful enemies with higher health and better accuracy etc. If the system detects multiple player deaths in quick succession or that the player is inaccurate and generally not playing well, the system lowers the players grade, and with that the enemies become fewer and less dangerous. The constant adjustment of difficulty without the player even knowing this system was in place, lead to a great experience for the players. Knowing that a game has dynamic difficulty adjustments in place can sometimes affect the experience for the players. In this way, a DDA system should adapt to the player in an inconspicuous manner, so the player is not aware of the outer intervention. Therefore, the system created in this study should perform in such a way as to be hidden from the player. While the techniques used in *Resident Evil 4* are bespoke to the game, the general idea of a player rating system to model ability is one that is prevalent in the field of DDA. The heuristics used would also be useful in the creation of a dynamic difficulty system.

2) Homeworld: One example of potential abuse of a difficulty system is seen in *Homeworld* [34]. The game is a space strategy simulator that involves the player collecting resources and building units to fight an AI team of similar units. At the end of a mission, the system would rate the power of your current fleet size and adjust the number of units that would be spawned when you started the next mission, effectively raising or lowering the difficulty. Unfortunately, once players realised this, they looked for ways to abuse the system by scrapping their entire fleet before ending the mission. This would make the next mission much easier since there would be fewer enemies spawned but by scrapping their fleet, they still retained the resources needed to quickly build another large and powerful fleet and 'steamroll' the enemy AI fleet. This kind of subversion of a video games systems is described in the book *Cheating: Gaining Advatage in Videogames* [35]. This book suggests there is no single reason people cheat in games.

However, using cheats to gain an advantage over another player appears to be the most egregious form of cheating. This is less likely to occur in games with dynamic difficulty since dynamic difficulty is most widely used in singleplayer games with AI opponents. Subverting the game system in a single player game can lead to an unintended Player Experience however and should be avoided when possible.

3) Left 4 Dead AI Director: The *Left 4 Dead* games created by Valve are 4 player cooperative first-person shooter games [36] [37]. The Artificial Intelligence (AI) director found within both *Left 4 Dead* games is focussed on maintaining pace: having an intensity that peaks and troughs throughout play [38]. This intensity fluctuates in real time and alters the frequency and strength of enemy waves during a given game. This keeps players on edge throughout the game and ensures the players make progress towards an end goal. If the players stay in any one area for too long, the director sends more enemies their way to encourage them to keep moving. This method makes the game far more replayable than games with more traditional, static spawning systems due to the uniquely unpredictable enemy spawns.

In order to determine each players 'skill' the AI director closely monitors the performance of each player. Determining a player's skill level quantitatively becomes problematic since we tend to judge skill based on qualitative measures. However, there are many attributes that could be employed to determine a players skill value in an fps: their overall health, damage ratio (received vs delivered), enemy kills, health packs used, shot accuracy, ammunition consumption rate and many more. These attributes can be used to help evaluate player skill and would be suitable heuristics to use in developing a dynamic difficulty system.

The AI director also attempts to model player emotional intensity, or 'stress' level, throughout each game. This helps determine how intense each player is finding the game at that time. This is quantitatively measured using factors such as: enemy proximity to player, number of enemies in level, number of enemies actively attacking player, enemy proximity to player on enemy death etc. In the game, special enemies have the ability to incapacitate the player, if this happens, the emotional intensity of the player reaches it's maximum value. As well as normal enemy spawns, the director also has the ability to spawn a 'mob' of 10 - 30 normal enemies behind or to the sides of the player route in order to surprise the player. There are limits to the number of enemies that can in the level at any one time and cooldowns to these "mob" spawn events so as not to completely overwhelm the players. The idea of using an emotional intensity value is very intriguing. It would be very interesting to see if this could be scaled down and adapted to suit a game with shorter levels.

Both player 'skill' and 'stress' values, along with other pieces of information such as how long the player has spent in the current area, are used by the AI director to manage the pace of the game. The director will influence the pacing by controlling where enemies are spawned, how many enemies are spawned, what enemies are spawned, how enemies will act, ammo and health spawns among many other factors. For example, a typical level will start with a small number of normal enemies, increase the spawn rates of enemies gradually and introduce special enemies until the players stress levels reach a maximum value. Then, at the peak of game intensity, the AI director will cease spawning enemies and give the players a brief respite of around 30 seconds to organize themselves before moving forward. Overall, this system to create pacing, peaks and troughs of intense gameplay would be an interesting route to go down to see if it could be adapted to suit shorter play times.

4) Hamlet: Hamlet [39] is a DDA tool built by *Hunicke and Chapman* that uses methods taken from *Operations Research and Inventory Management*. The tool would adjust the difficulty of the first person shooter *Half-Life* [40] as the player progressed through the level. Algorithms were created to measure the difficulty based on the speed at which the player progressed and what resources the player currently had. This included functions for:

- *Monitoring game statistics*
- *Defining adjustment actions and policies*
- *Executing those actions and policies*
- *Displaying data and system control settings*

The system essentially manipulates the game difficulty by studying the player's actions, predicting when the player no longer has the means to continue progressing and adjusting the supply and demand of the players inventory.

IV. Hypotheses

When designing a dynamic difficulty system there are several objectives one can hope to achieve. One objective is to create a system that leads to increased replayability of a game. One might also wish to implement DDA to make a game more accessible to players of varying skill levels. Another aim could be to create a system that enhances overall player enjoyment. For the scope of this paper, the focus is on creating a dynamic system that is effective in a game with shorter levels. Effective can mean a variety of things, in this case an effective dynamic difficulty system would be one that is more enjoyable and engaging than that of a standard static difficulty system and also has less variation in player performance or challenge perception. Xue et al [41] found that when DDA techniques were applied to multiple mobile games, player engagement increased by up to 9 percent. These findings suggest that dynamic difficulty could increase player engagement in other games. While engagement encompasses multiple factors, This paper attempts to look at enjoyment specifically. Therefore the first hypothesis is that players will find playing the dynamic difficulty version of the game more fun than when playing the static difficulty.

The second hypothesis considers the fact that in the literature, dynamic difficulty adjustments are said to help solve the problem of scaling difficulty for different players with different abilities [42] [39]. In that regard, a well designed dynamic difficulty setting that takes individual player skill as a variable should, in theory, be perceived as equally challenging to all players, whereas a static difficulty system should be perceived as more challenging to less skilled players and less challenging to more skilled players. Therefore the second hypothesis of this

TABLE I
HYPOTHESES TABLE

	Hypothesis	**Null Hypothesis**	Data Source
1	Players will enjoy playing the version with dynamic difficulty more than the version with static difficulty	There is no significant difference in levels of enjoyment between the static version and the dynamic version	Questionnaire
2	There will be more variation in how challenge was perceived with the static version than with the dynamic version	There is no significant difference in variation of challenge perception between the static version and the dynamic version	Questionnaire
3	There will be more variation in player performance with the static version than with the dynamic version	There is no significant difference in variation of player performance between the static version and the dynamic version	Gameplay Statistics
4	Players will be able to reliably judge which version used dynamic difficulty	Players will be unable to reliably judge which version used dynamic difficulty	Questionnaire

paper is that there will be less variance in the challenge that different players perceive when using the dynamic difficulty setting compared with the static difficulty setting.

The third hypothesis follows directly from the second. Rather than dealing with how challenge is perceived by players, this hypothesis looks at the actual performance of the players. As stated in the previous hypothesis, players should be challenged more equally by the dynamic version of the game. Therefore this hypothesis states that there will be less variance in the performance of players when using the dynamic difficulty setting compared with the static difficulty setting. This will be assessed using each player's kill/death ratio for each system.

The fourth Hypothesis is based on the belief that players will not be able to identify which level used a dynamic difficulty system. Research suggests that a good DDA system should be hidden from the player and that most players should not realise that they are playing a game which uses dynamic difficulty lest they feel cheated [39]. Therefore, if the DDA Algorithm is performing adequately, it would be important to see if players can tell if dynamic difficulty is being used.

V. METHODOLOGY

The philosophy underpinning this paper is largely based on Empiricism. The view that information is primarily gained through sensory experience. Hence my methodology will mainly talk about how to quantify subjective data using likert scales and using telemetry data in order to empirically evaluate the effectiveness of my computing artefact.

A. Sample Group

In order to acquire the required sample size within a short timeframe, The Games Academy studio at Falmouth University was the most convenient locale to gather players for research. Students are selected by convenience, with the researcher hoping for a random range of gaming abilities, although it must be noted that the proportion of experienced gamers is likely to be higher since the Games Academy consists entirely of game development students. In order to account for this possiblity, the participants are questioned on their prior gaming experience so that experience can be discussed with regards to the results. Future studies would benefit from a more diverse selection of participants but due to time and funding limitations it was deemed as a necessary limitation. Three *a priori* power analyses were conducted for this experiment. Using the Bonferonni Correction [43] method the alpha value of $\alpha = 0.05$ was divided by 4 given that I will be testing 4 hypotheses to give alpha values of $\alpha = 0.0125$. The first analysis consisted of a one tail T-test with an expected effect size of $d = 0.5$, a Bonferonni-corrected alpha error probability of $\alpha = 0.0125$ and a power of $1 - \beta = 0.95$. The resulting recommended sample size was $N = 59$. The second power analysis was applicable to both the second and third hypotheses and consisted of a one tail F-test of variance with an expected ratio variance of 3, a Bonferonni-corrected alpha error probability of $\alpha = 0.0125$ and a power of $1 - \beta = 0.95$. The resulting recommended sample size was $N = 50$. The final power analysis consisted of a Chi-Square test with an expected effect size of $d = 0.6$, a Bonferonni-corrected alpha error probability of $\alpha = 0.0125$ and a power of $1 - \beta = 0.95$. The resulting recommended sample size was $N = 46$. Therefore the minimum number of samples required is $N = 59$. The sample actually acquired during this study was $N = 61$.

The sample consists of one group of players, each player playtested both difficulty settings during the experiment. Each player will be given a consent form as well as an information sheet with only vital information such as control layouts, so as to limit bias. Each playtest will last no longer than 3 minutes per level. This means that each experiment should last between 10 to 20 minutes depending on the skill of the player.

B. Playtest Session

A diagram of the experiment process for each participant can be seen in *figure 22*. The playtesting of the game requires each player to play through a short tutorial to begin with. This tutorial will involve completing several tasks. The first task will be traverse through a set of short waypoints. The next task will be to fire on and defeat several stationary enemy targets. A task will also be given to use any special abilities the player may have. This brief tutorial should help the player to become acclimated to the basic controls of the game as well as the general game systems. This will help to limit the practice effect the player might experience later on during testing of the two systems.

Once the tutorial is finished, the game will randomly load one of the two level versions. One level version will include the static difficulty setting, the other will include the dynamic difficulty setting. The difficulty setting that the player is

testing will be unknown to both the researcher and the player. The double-blind nature of the research will help to reduce observer bias as shown in Hrbjartsson *et al* [44].

The player will have 2 minutes 30 seconds to complete the level. This will be unknown to the player since this may affect gameplay decisions by introducing unwanted pressure bias. This completion time will be measured by the game. If the player completes the level within the time limit, the completion time, number of kills and number of deaths will be collected automatically and an in-game questionnaire will appear asking the player questions about their experience playing the level. If the player does not complete the level within the allotted 10 minutes then the level will end automatically as if the player had completed the level, but the game will indicate in the statistics gathered that the player did not finish the level. Once the player has finished the first version, they will then play the alternate version, following the same process as the first. The anonymous results of both playtests will be automatically sent to a secure university network drive.

Questionnaires are the most widely used research method, consisting of a well defined set of questions to which an individual is asked to respond [45]. The quality of the questions in a questionnaire correlates directly to the usefulness of a questionnaire, as ill-defined or vague questions can result in bad data [46]. Questionnaires can also be subject to confirmation bias; if a question written by the researcher about an experience seems loaded rather than being as neutral as possible then the player response won't be as useful, skewing the results in favour of the researchers preconceptions about the experience [47]. This is a common pitfall with interview and focus group questions as well. A web-based questionnaire has the ability to disseminate the test to large swathes of a population regardless of geographical location and gather user responses relatively quickly and are noted to be preferred over paper questionnaires [48].

In the questionnaire used in this experiment players are asked whether or not they agree with each statement ranging from 0 (not at all) to 100 (extremely). The questions were originally set to a 5 point likert-scale as is used in the Core module questions in the Game Experience Questionnaire (GEQ) [9] [49] and how the data was gathered in Polymorph as discussed previously [24]. However, in order to gather the most detailed information, I will be gathering ratio data from 0 to 100 as ratio scaling provides the most detailed and precise information quantitatively. It could be argued that people cannot discriminate much beyond a 7 point scale, however I believe a percentage scale will be useful as people have a familiarity with percentage ratings, hence why they are sometimes used in the 'Feeling Thermometer' and other such tools. Since I am trying to gauge player feelings and agreements I believe it is an appropriate scale to use in my experiment questionnaire. The GEQ includes seven components:

1) *Imaginative and sensory immersion*
2) *Competence*
3) *Flow*
4) *Tension*
5) *Positive affect - fun and enjoyment*
6) *Negative affect*
7) *Challenge/Suspense*

Since immersion is mainly considering the games story and aesthetics the researcher has elected to not include questions about this component. Each statement in the GEQ matches one of these Player Experience Components. There are several statements relating to each component and the order in which the statements/questions are placed in the questionnaire is mixed. The mean score of each components results are then used to inform the results on that particular part of the player experience. This is to limit acquiescence response bias and to get more realistic and correct data [50]. The majority of the testers are students, taking time out of their day to playtest the game and answer the questionnaire. Ideally the questionnaire should therefore be much shorter than 27 questions, especially if they must play both versions of the game and fill out a questionnaire for each. Therefore the questions have been altered and narrowed down to the questions in *figure 10*. Clearly, under other circumstances, more questions would provide more reliable results, but it has been shown that if questionnaires are too long they can cause the user to stop answering thoughtfully which can skew results and introduce bad data. Under the conditions of the experiment and with whom will be participating in the playtests, it is best to be prudent and not draw out the tests to interminable lengths.

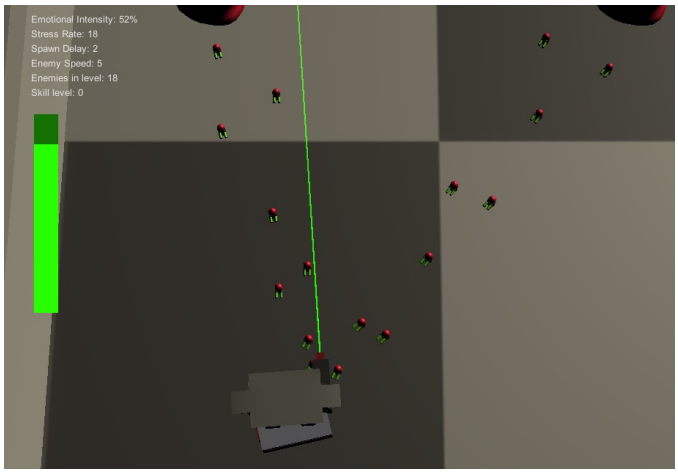

Fig. 2. Screenshot of the initial prototype Artefact

C. Ethical Considerations

There is a legal right for participants in research studies to be fully informed and consent must be explicitly given before participation in any experimentation. These rights are grounded in international law from The Nuremburg Code established in 1947 and Helsinki Accords established in 1975. There should be little to no ethical risks involved in this study. The researcher is continually referencing the British Computer Society Code of Conduct [51] and its guidance on duty and integrity when conducting any studies and considering any ethical implications. Participants have been sampled from members of the Games Academy and were subject to the Code of Research Ethics at Falmouth University. Each participant read and agreed to a consent form (see *figure 9*) as well as

an information sheet discussing the reasons for the study. It was stated that all participants had the right to opt-out of the study any any time without consequences and will have all the rights afforded to them as stated in the General Data Protection Regulation [52] if applicable, however, personal information such as name, date of birth, gender were not needed in this study. All data gathered was anonymous and confidential, with any data being stored on a secure university network drive. Participants were asked to pause the study and take a break in the unlikely event that the experiment lasted more than 30 minutes. This was to prevent any possible bouts of nausea, eye strain, or general discomfort from looking at computer screens for too long. The software was designed to account for this possibility. A risk assessment was conducted and the precautions were taken such as the limitation of the experiment to under 30 minutes. Average playtime is expected to be 15 minutes.

VI. Artefact

The computing artefact is a top-down 2D shooting game wherein the player defends themselves from an unending wave of zombie enemies. A table of user stories used in development can be found in appendix B. The following subsections describe the game in more detail.

A. Basic Mechanics

The player has several abilities or mechanics in-built. The player has eight-directional movement using the WASD keys and can use the mouse to aim the cursor and thus aim the gun. The player has a cone of vision as shown in *figure 4* that aims in the direction between the player character and cursor. This aiming cone shrinks and expands based on movement. The cone expands to a maximum of 25 degrees when the player moves to simulate less accuracy as is used in most first person shooters. The cone shrinks to a minimum of 5 degrees when the player is not moving or when the player 'aims down the sights' (ADS) using the right click mouse button or Left Shift, however the player movement speed is altered by a factor of 0.7, thus slowing the player and making them more vulnerable. The player can shoot the gun using the left click mouse button. On click, a random vector within the aiming cone field of view is chosen and a ray is cast in that direction to simulate the bullet. It takes three shots to kill an enemy when hit in a not vital area. The center of the front of the enemy is considered a vital area, therefore when the player bullet hits that weak spot the enemy is automatically killed regardless of it's remaining health. This introduces a further element of skill to the game. The gun has a magazine of 12 bullets and reloads automatically when empty but can be reloaded manually by pressing the 'R' key. This adds another strategic element to gameplay.

B. Tutorial

The Artefact begins with the consent form displayed (see *figure 9*). Once the player ticks the consent icon and presses Enter, the tutorial begins. The tutorial, as seen in *figure 4*, has a list of objectives in the top left of the screen and contains pop-ups in the bottom center that guide the player on their current objective.

The tutorial first teaches the players to move through a short bended corridor. Then players are taught how to shoot using stationary targets. Next the ADS mechanic is introduced and promoted as the players are given the objective to shoot the next batch of targets whilst the player is moving. These target can only be damaged if the player is moving and shooting. This is essential to teach the player since doing well at the game relies heavily on the player being able to move and shoot at the same time. Then the player moves on to the next area and has to contend with a few normal enemies that can actually attack the player in order to acclimatise the player to the basic idea of the game.

Initially, the user interface for the tutorial pop-ups were all the same colour but after pilot tests were carried out, it was noted that players did not realise the pop-ups were changing and so were not re-reading the next prompts. Therefore each time the pop-up changed, the colour scheme was changed to make it clear that new information was being given to the player. Also, the main controls were added to the top right of the tutorial to help less experienced players if they forgot the controls.

C. Main Game

The main game level layout (see *figure 3*) is designed to encourage constant movement of the player. The simple placement of the 6 equally spaced out broken windows (enemy spawn areas) around the play space along with slightly different base spawn rates makes where the next enemy is coming from seem unpredictable to the player. This makes the center of the play space seem the safest to the player since they have a nearly full view of the level (see *figure 20*). The placement of the inner walls was used to limit line of sight and create choke points to make the game more difficult for the player and promote player movement. This has the added effect of grouping enemies chasing the player to become swarms that if left unchecked can overwhelm the player.

The various furniture objects in the level are there for aesthetic effect only and have no collision with the player or enemies. This is explained in the tutorial so as not to confuse the player. The only objects that the player collides with are the walls, broken windows and enemies.

VII. Artefact Development

The Artefact was built using Unity 2018.4.2f1[1]. The DDA Algorithm was written in C# using Visual Studio 2019[2] as was the Top Down Shooting Game that the Algorithm would be used in.

The enemies move using the free system "A* Pathfinding Project"[3] found on the Unity Asset Store.

[1] https://unity.com/
[2] https://visualstudio.microsoft.com/vs/
[3] https://arongranberg.com/astar/

Fig. 3. Screenshot of the Level Layout with Spawn Points Circled Red

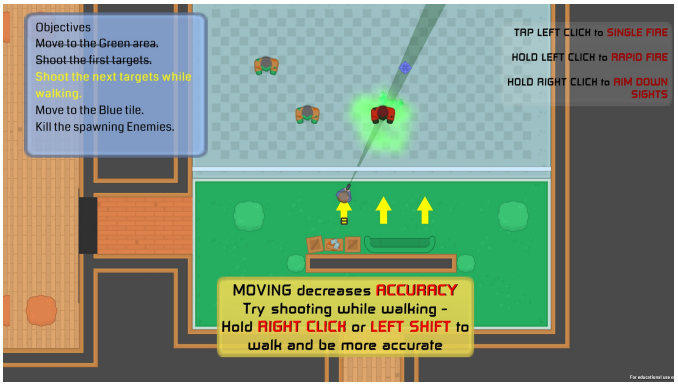

Fig. 4. Screenshot of the Tutorial

The game art was procured for free using kenney assets[4]. The music played during the game is called fiberitron and is a royalty free track found online[5]. The sound effects used are all free sounds found online[6]. The following subsections describe the Artefact and how it was developed.

A. Dynamic Difficulty Algorithm Development

Initially the plan was to use an AI Director and player grading system similar to those used in *Left 4 Dead* [36] and *Resident Evil 4* [33]. This would involve monitoring and altering heuristics as the game is being played. Originally the Artefact used a modified version of the group team game as a base (see *figure 3*). This included a twin stick player character, enemy spawners, and enemies. A script was created called AIDirector.cs that altered spawn rates and enemy statistics such as speed, based on an emotional intensity value that ranges from 0 to 100. This value increased or decreased based on a number of factors, including but not limited to player skill, player health, number of enemies and proximity of enemies.

During early playtesting of this system, it became clear that the complex nature of the Algorithm lead to unsatisfactory results. In other words, the game was far too simple for such a complex set of heuristics to come together and amount to desirable outcomes. Having so many factors affecting just the spawn rate and enemy speed meant that fine tuning the effect each individual heuristic had on the difficulty was very challenging. Therefore it was decided that the algorithm should be simplified. This lead to the final algorithm which is fairly simple in essence.

The minimum and maximum values for spawn rate and enemy speed were set and adjusted based on testing using an AI agent and pilot testing. This is discussed in the Software Testing section of the paper. Pseudo code of the dynamic algorithm can be seen in *figure 23*and the static version in *figure 24*. The enemies had a minimum speed of 180 and a maximum of 320. The spawners had a minimum delay of 2 seconds and a maximum delay of 10 seconds. In the dynamic functions the spawn delay modifier was 0.2 and enemy speed modifier was 2.

B. Final Dynamic Difficulty Algorithm

The final design works as follows. The enemy movement speed has a value. This value is increased by a certain factor each time an enemy is killed, thus making the game more and more difficult. If the player dies, the amount by which the enemy movement speed has increased is then halved. This lowers the difficulty. If the player dies again, that increase from the base enemy speed value is halved once again, thus lowering the difficulty again. This same method is used to increase or decrease the enemy spawn rate. This is a far more simple algorithm than the previous design but works much more efficiently at simulating dynamic difficulty in order to match the skill of the player. Design adjustments were made to accommodate the new design, player health was reduced so that the player would be killed in one hit for example. This meant that it would be easier to judge player skill and challenge based on the telemetry data being gathered during testing. See *figure 23* for pseudo code example.

C. Final Static Difficulty Algorithm

The static difficulty algorithm was designed so that difficulty would start off easy and steadily increase as the game went on. This is to replicate the steady increase in difficulty found in most commercial video games. The game begins with base spawn rates in the AI Director starting at the same minimum value as the dynamic system does, as does the enemy speed. Since these are the only two values that are changed in the dynamic system, it was decided that this is how difficulty should be altered in the static system as well. The spawn rate starts at the minimum value and increases gradually over time until it eventually reaches the maximum spawn rate once the level timer is finished. This increase is linear and proportional to the level timer. The same process is carried out with the enemy speed value. See *figure 24* for pseudo code example.

D. System Development Life Cycle

The software used in this experiment was developed in it's entirety by one person. The initial prototype was created in

[4]https://kenney.nl/assets/topdown-shooter
[5]https://www.dl-sounds.com/royalty-free/fiberitron-loop/
[6]https://www.zapsplat.com/sound-effect-categories/

conjunction with a video game that was in development by a team of developers including the Researcher, however, it was decided early on to discontinue this and to develop an entirely bespoke game for the project. This is mainly due to the potential risks involved in having to depend on the group based project reaching certain development milestones and not blocking the Researcher's continued development of the Artefact. Also, designing a bespoke DDA system was deemed to be much easier if the Researcher had full control of the design decisions that went into designing the game that it was to be used with. The first version of the bespoke game can be seen in *figure 19*.

When developing the Artefact, the initial life cycle method proposed was the Agile Method, since team collaboration and communication would have been essential. After the decision to create a bespoke game was made, however, the development life cycle that was most suited to the project became The Prototyping Model [53]. This is because the Agile Life Cycle Model has steps that were no longer applicable to my project. For example, communication is vital when working with a team on a project but was now unnecessary since the group project was no longer the basis of the research. Likewise, principles such as self-organizing teams, regular discussions on team improvement and face-to-face conversation with teammates were no longer appropriate for my project. An experimental approach to The Prototyping Model was most suited to the project since, like Agile, it is an iterative process that is more responsive to user needs. The Prototype Model seemed much more in line with the requirements of the project since the DDA system would require a lot of testing with many working versions, all requiring user-feedback in order to develop subsequent prototype versions. This is the essence of the Prototyping Life Cycle.

In order to create the Dynamic Difficulty Algorithm, the Artefact was iteratively designed and playtests were conducted throughout development in order to validate and verify it [54]. Therefore it was ensured that anyone who playtested throughout development was not in the final experiment, since this may have biased the results. It was decided early on that creating an AI agent would be more beneficial and more efficient than gathering playtesters constantly throughout development and so a Player AI was designed for testing purposes.

The user needs, such as the flow of the game and the effectiveness of the DDA Algorithm could be proposed and evaluated through pilot testing and AI simulations. Then once feedback is attained, the system could be refined to attempt to improve and revise the Prototype to then be tested again until the Prototype satisfies all user requirements.

E. Software Testing

The DDA Algorithm functionality was tested using a combination of pilot testing and a Player Artificial Intelligence Agent that simulates the player. Initially, each iteration of the Artefact was to be tested using human players. However, this seemed to be an impractical method of testing the DDA Algorithm. The development of a working dynamic difficulty system requires many refinements and iterations to be tested, which would mean a lot of participants would be required to do pilot tests throughout development. There are ways of limiting the impracticality of this, such as with posting the Artefact online to get feedback from players, on the other hand, these methods also come with drawbacks and would likely slow development considerably while waiting for feedback. Thus, a Player AI was developed to simulate the player so that testing of the DDA Algorithm could be much faster.

TABLE II
PLAYER AI VERIFICATION RESULTS

Human Pilot Test		
Kills	Deaths	KDR
132	4	33
132	10	13.2
110	14	7.9
138	3	46
125	6	20.8

Average KDR
24.2

Player AI Test		
Kills	Deaths	KDR
137	6	22.8
141	10	14.1
131	2	65.5
135	6	22.5
137	13	10.5
141	7	20.1
137	7	19.6
138	5	27.6
135	5	27
138	6	23

Average KDR
25.3

The Player AI has fairly basic pathfinding abilities and as such cannot be said to be as human-like as can be but serves the purposes of testing the difficulty system. The Player AI moves between the four corners of the map using the same A* Pathfinding that the enemies in the game use. The Player AI will interrupt it's pathfinding and move away from enemies that get too close however, somewhat simulating a human player. The Player AI has most of the same systems as the Player Prefab in-game. To adjust the Skill Level of the Player AI, the shots per second and accuracy values can be adjusted. As the player has an accuracy cone system built into the design, the Player AI uses this when aiming at the nearest enemy with a clear line of sight and therefore the Skill Level of the AI can be adjusted by adjusting the spread of this cone, thus adjusting the shot accuracy of the AI. Pilot tests were carried out to assess the Player AI against human players and adjust the shots per second and accuracy values in order to get as accurate a representation of human gameplay as possible. A static difficulty curve with an unchanging spawn delay of 5 seconds and enemy speed of 330 was used in the pilot test. The average Kill/Death Ratios (KDR) of the five pilot testers was used as a base to aim towards when adjusting the shot accuracy and shooting rate of the Player AI. When the Player AI had achieved a similar enough KDR in simulations with the same static difficulty, it was deemed as being satisfactorily functional for the purposes of testing the DDA Algorithm. The results of the pilot test and final Player AI tests can be seen in Table II.

VIII. RESULTS AND ANALYSIS

The data gathered from the experiment were output into a csv file on the university network drive. RStudio 1.2.5019[7]

[7]https://rstudio.com/products/rstudio/download/

was used to collate and process the results. The R package ggplot2 was used to generate the graphs. See Appendix B for the code used to analyse the data.

A. Hypothesis 1 - Players will enjoy playing the version with dynamic difficulty more than the version with static difficulty

This hypothesis was intended to judge whether or not players enjoyed playing the dynamic version of the game over the static version. *Figure 6* shows us that there is a slight preference for the dynamic version. A t-test was carried out analysing the differences in how the players scored the two difficulty versions on the basis of player enjoyment. This test produced a p-value of $p = 0.001638$ and a t-value of $t = 3.0634$. The p-value is very low and therefore the null hypothesis can be rejected. The mean value of the differences was 6.08 showing that on average players enjoyed the dynamic version 6 percent more than the static version.

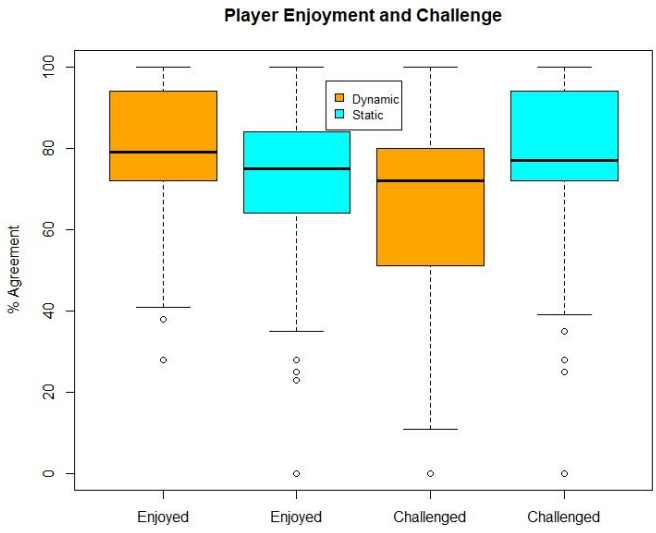

Fig. 5. Boxplot of Enjoyment and Challenge Perceptions

B. Hypothesis 2 - There will be more variation in how challenge was perceived with the static version than with the dynamic version

Interestingly, player perceptions of challenge may not be as varied based on the results of "I felt challenged" in the questionnaire. Therefore an F-Test was carried out in order to determine the variance in the challenge perception scores received by the static version when compared to the results of the dynamic version. This test analyses player perception of challenge. This test produced a p-value of $p = 0.4927$. This p-value is too high and therefore the null hypothesis would not be rejected. The ratio of variances was 1.0047.

This seems to indicate that although the actual challenge appears to be more varied based on player deaths with the static difficulty, the perception of challenge among players is less varied in the static version when compared to the dynamic version than one would expect.

Report an effect size ——————————
¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡

C. Hypothesis 3 - There will be more variation in player performance with the static version than with the dynamic version

An F-Test was carried out to determine the variance of player deaths in the static version compared with the variance of player deaths in the dynamic version. This is deemed by the researcher to be a valid assessment of actual challenge. The test produced a p-value of less than 0.0001. This p-value is extremely low and therefore the null hypothesis can be rejected. The ratio of variances was 4.9226. This means that there is more variance in the number of deaths players suffered in the static version. This leads one to suspect that with a diverse range of player skill levels, a static difficulty system is either too challenging or too easy for more players than would be the case in a dynamic difficulty system.

FIX HYPOTHESES

D. Hypothesis 4 - Players will be able to reliably judge which version used dynamic difficulty

The data showed that 36 out of the 61 participants in the study correctly guessed which level used the dynamic difficulty system. A Chi-squared test was performed to determine the probability that this was higher that random chance. The test produced a p-value of $p = 0.159$. This p-value is too high and so the hypothesis that suggests players would be able to predict which version used dynamic difficulty cannot be said to be true. This does indicate that these results occurred at a rate closer to random chance, thus suggesting that the null hypothesis cannot be rejected and that players were unable to reliably identify which system used dynamic difficulty.

IX. Discussion

The results from this experiment suggest that player enjoyment and challenge is affected by a games difficulty system. Although not drastically different, as shown in *figure 5*, the evidence suggests that the dynamic difficulty as implemented in this particular experiment was more enjoyable and less challenging to players than the static difficulty. The potential reasons behind this are discussed in this section.

The results indicate that players were unable to reliably identify the level that used dynamic difficulty. This adds to the validity of player perception results since it's unlikely that players felt cheated by the dynamic system if they were not aware of it.

As you can see from the findings illustrated in *figure 6*, on average players found the dynamic version more enjoyable than the static version. This is a fairly small difference but since the difficulty system is the only thing that was changed between the two levels, it's still a significant difference. The findings also show that players generally didn't get bored playing either version.

The static version could become too hard for some players to enjoy whereas the dynamic version suffered from this

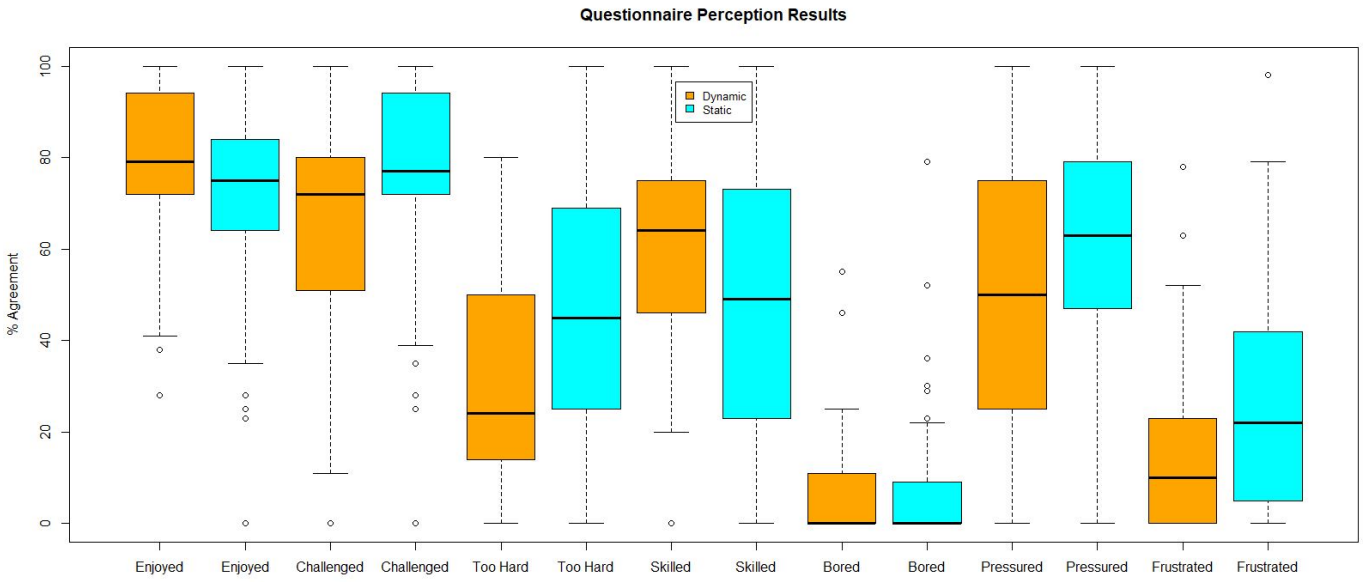

Fig. 6. How Players Felt Playing Each Difficulty System

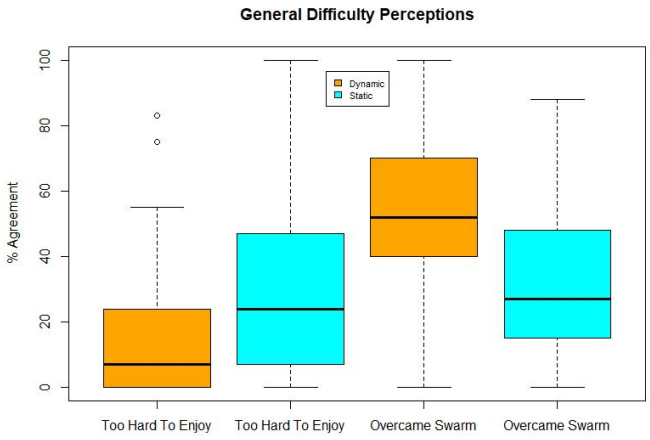

Fig. 7. Boxplot of Challenge Perceptions

problem less. Evidence that this could be the case can be seen in *figure 7*. This may indicate that the dynamic algorithm is functioning correctly in adapting challenge to player skill. The alternative could be that the static version was just designed to be too difficult for most players despite the researcher using the pilot tests to verify the overall difficulty of the static version.

The game would allow up to 15 enemies on the screen at a time if players could not defeat them fast enough, this could lead to a swarm of enemies attacking the player at times. As shown in *figure 7*, these swarms were more likely to be overcome by the player in the dynamic version. This is likely due to the fact that enemy speed values would be lowered if the player is struggling in the dynamic version whereas in the static version the enemy speed and spawn rate steadily increases regardless of the skill of the individual player. The ability of the player to overcome adversity is a possible factor in how much they enjoyed the game and how difficult it felt for them.

In *figure 15* we can see that with the dynamic version, there is a very slight positive correlation between how skilled players felt and how challenged they felt. The opposite can be seen with the static version in *figure 16* where it seems that the more skilled a player felt, the less challenging the game felt. This is an interesting observation and could suggest that when a difficulty system is dynamic, the player is more likely to feel like they are good at the game despite feeling that the game is more challenging to them or perhaps even because the game is challenging them appropriately for how good they are.

In *figure 6* we see that with the dynamic version, players never felt bored and rarely felt frustrated. This, coupled with the fact players did feel challenged and similarly skilled, promotes the idea that the dynamic version successfully achieved 'Game Flow' as discussed previously and in *figure 1*. This cannot be said for the static version since although not feeling bored, more players felt frustrated and there was much more disparity between how challenged they felt and how skilled they felt.

Contextualize findings in lit review

X. Limitations

Although it is particularly encouraging to see that the results in *figure 6* reflect the idea that the dynamic version indeed created flow, it cannot be forgotten that flow is particularly difficult to quantify and correctly assess. This is especially true with a game as short as the game in this experiment. Replicating the Player Experience of a commercial game in a study like this is very difficult. The external validity of the experiment could be improved in a way that is not possible in this study if the researcher had more time and resources at

hand. Perhaps modifying an existing commercial game would produce better insights into true Player Experience.

It must be noted that one player decided to go against the objective of killing as many enemies as possible and just outrun the enemies in the dynamic version of the game. Since the enemies were not being killed and there was a limit of 15 enemies spawned at a time, the player was able to outrun the enemies for the whole 150 seconds without dying. This is a design flaw in the algorithm the researcher did not foresee. It should be noted that this player was a professional game designer and the data from this player was omitted from the rest of the results since it would not be fairly judging the algorithm as it's intended to work. This type of game system subversion was described earlier in the paper when discussing the dynamic difficulty in *Homeworld*. To correct this, the algorithm could be adjusted to increase the difficulty when a stalemate like this occurs.

The Player AI created and used for testing was far from perfect and given more time and resources, a much better representation could have been created. Ideally an AI would not have been needed and more pilot tests with actual human players would have been possible.

Due to the advent of the Covid-19 global pandemic, certain restrictions and unfortunate limitations were necessary towards the end of this study. Social distancing was introduced and the University was closed. This made access to certain resources and support more challenging than would have been preferred.

XI. Further Research

As stated in the limitations section above, there are many things that could be studied further to gain insight into how dynamic difficulty can affect player enjoyment and challenge. Player experience and game flow are particularly hard to quantify and assess in such a small scoped experiment. A more lengthy study on a game with a longer experiment time would be ideal. Since most commercial games can last several hours with a much more content, it is important to be able to conduct more experiments on dynamic difficulty in a controlled manner on a much larger scale than was done in this study. The levels in this study only lasted two and a half minutes each and so cannot replicate the experience created by most commercial games today. It would be interesting to see how bigger games with more content would be affected by dynamic difficulty compared with standard static difficulty systems. More research on dynamic difficulty systems used on existing commercial games compared with the static systems those commercial games were originally designed with would also be interesting.

XII. Conclusion

Although limited in scope, from this experiment we can see how a DDA system can be more effective at creating a better player experience than a static difficulty system. We can conclude that in the case of this experiment Dynamic Difficulty is more enjoyable than a Set Static difficulty. This is likely due to how the dynamic system adapts to the skill of the player and keeps the player in the 'Flow Zone' (see figure 1). The results show us that more players were frustrated by the Static difficulty version but very few were bored by either version. This suggests that the Dynamic version was better at achieving a state of Flow than the Static version. This is evidenced further by the fact that player performance is more diverse in the Static Difficulty, with player death values being more highly varied amongst players. It seems that finding ways to implement DDA into larger scoped commercial games would be a worthwhile endeavour for developers, and would enhance the gameplay experience for players of all skill levels, leading to a wider market for those games.

XIII. Acknowledgements

I would like to thank all of the staff at Falmouth University for their help and support, particularly on the BSc Computing for Games Course. I would also like to thank my dissertation supervisor Brian McDonald for keeping us students on track. Many thanks also to my fellow university students for their help with playtesting and dissertation feedback.

REFERENCES

[1] J. Juul, *A casual revolution: Reinventing video games and their players*. MIT press, 2010.

[2] C. Liu, P. Agrawal, N. Sarkar, and S. Chen, "Dynamic difficulty adjustment in computer games through real-time anxiety-based affective feedback," *International Journal of Human-Computer Interaction*, vol. 25, no. 6, pp. 506–529, 2009.

[3] A. Denisova and P. Cairns, "Adaptation in digital games: the effect of challenge adjustment on player performance and experience," in *Proceedings of the 2015 Annual Symposium on Computer-Human Interaction in Play*. ACM, 2015, pp. 97–101.

[4] K. Isbister and N. Schaffer, *Game usability: Advancing the player experience*. CRC press, 2008.

[5] M. Czikszentmihalyi, *Flow: The psychology of optimal experience*. New York: Harper & Row, 1990.

[6] A. McMahan, "Immersion, engagement, and presence: A method for analyzing 3-d video games," in *The video game theory reader*. Routledge, 2013, pp. 89–108.

[7] K. Mikami, K. Kondo *et al.*, "Adaptable game experience based on player's performance and eeg," in *2017 Nicograph International (NicoInt)*. IEEE, 2017, pp. 1–8.

[8] M. Ambinder, "Biofeedback in gameplay: How valve measures physiology to enhance gaming experience," in *game developers conference*, vol. 2011, 2011.

[9] W. IJsselsteijn, Y. De Kort, and K. Poels, "The game experience questionnaire," *Eindhoven: Technische Universiteit Eindhoven*, 2013.

[10] M. Csikszentmihalyi, "Toward a psychology of optimal experience," in *Flow and the foundations of positive psychology*. Springer, 2014, pp. 209–226.

[11] K. M. Gilleade and A. Dix, "Using frustration in the design of adaptive videogames," in *Proceedings of the 2004 ACM SIGCHI International Conference on Advances in computer entertainment technology*. ACM, 2004, pp. 228–232.

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

[13] R. Koster, *Theory of fun for game design*. " O'Reilly Media, Inc.", 2013.

[14] J. T. Alexander, J. Sear, and A. Oikonomou, "An investigation of the effects of game difficulty on player enjoyment," *Entertainment computing*, vol. 4, no. 1, pp. 53–62, 2013.

[15] H. Qin, P.-L. P. Rau, and G. Salvendy, "Effects of different scenarios of game difficulty on player immersion," *Interacting with Computers*, vol. 22, no. 3, pp. 230–239, 2009.

[16] J. Chen, "Flow in games (and everything else)," *Commun. ACM*, vol. 50, no. 4, pp. 31–34, Apr. 2007. [Online]. Available: http://doi.acm.org/10.1145/1232743.1232769

[17] K. Salen, K. S. Tekinbaş, and E. Zimmerman, *Rules of play: Game design fundamentals*. MIT press, 2004.

[18] R. Bartle, "Hearts, clubs, diamonds, spades: Players who suit muds," *Journal of MUD research*, vol. 1, no. 1, p. 19, 1996.

[19] G. N. Yannakakis, P. Spronck, D. Loiacono, and E. André, "Player modeling." Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2013.

[20] J. Preece, Y. Rogers, H. Sharp, D. Benyon, S. Holland, and T. Carey, *Human-computer interaction*. Addison-Wesley Longman Ltd., 1994.

[21] G. Fischer, "User modeling in human–computer interaction," *User modeling and user-adapted interaction*, vol. 11, no. 1-2, pp. 65–86, 2001.

[22] G. N. Yannakakis, "Game ai revisited," in *Proceedings of the 9th conference on Computing Frontiers*. ACM, 2012, pp. 285–292.

[23] C. Pedersen, J. Togelius, and G. N. Yannakakis, "Modeling player experience in super mario bros," in *2009 IEEE Symposium on Computational Intelligence and Games*. IEEE, 2009, pp. 132–139.

[24] M. Jennings-Teats, G. Smith, and N. Wardrip-Fruin, "Polymorph: dynamic difficulty adjustment through level generation," in *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*. ACM, 2010, p. 11.

[25] M. Zohaib, "Dynamic difficulty adjustment (dda) in computer games: A review," *Advances in Human-Computer Interaction*, vol. 2018, 2018.

[26] D. Wheat, "Dynamically adjusting game-play in 2d platformers using procedural level generation," 2013.

[27] P. Spronck, I. Sprinkhuizen-Kuyper, and E. Postma, "Online adaptation of game opponent ai with dynamic scripting," *International Journal of Intelligent Games and Simulation*, vol. 3, no. 1, pp. 45–53, 2004.

[28] L. Lidén, "Artificial stupidity: The art of intentional mistakes," *AI game programming wisdom*, vol. 2, pp. 41–48, 2003.

[29] S. Woodcock, "The future of game ai: A personal view," *Game Developer Magazine*, vol. 7, no. 8, 2000.

[30] S. Rabin, "Promising game ai techniques," *AI Game Programming Wisdom*, vol. 2, pp. 15–27, 2004.

[31] P. Demasi and J. d. O. Adriano, "On-line coevolution for action games," *International Journal of Intelligent Games & Simulation*, vol. 2, no. 2, 2003.

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

[33] "Resident Evil 4," Capcom, 2005.

[34] "Homeworld," Sierra Studios, 1999.

[35] M. Consalvo, *Cheating: Gaining advantage in videogames*. Mit Press, 2009.

[36] "Left 4 Dead," Valve, 2008.

[37] "Left 4 Dead 2," Valve, 2009.

[38] M. Booth, "The ai systems of left 4 dead," in *Artificial Intelligence and Interactive Digital Entertainment Conference at Stanford, 2009*, 2009.

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

[40] "Half-Life," Sierra Studios, 1998.

[41] S. Xue, M. Wu, J. Kolen, N. Aghdaie, and K. A. Zaman, "Dynamic difficulty adjustment for maximized engagement in digital games," in *Proceedings of the 26th International Conference on World Wide Web Companion*. International World Wide Web Conferences Steering Committee, 2017, pp. 465–471.

[42] O. Missura and T. Gärtner, "Player modeling for intelligent difficulty adjustment," in *International Conference on Discovery Science*. Springer, 2009, pp. 197–211.

[43] E. W. Weisstein, "Bonferroni correction," *https://mathworld. wolfram. com/*, 2004.

[44] A. Hróbjartsson, A. S. S. Thomsen, F. Emanuelsson, B. Tendal, J. Hilden, I. Boutron, P. Ravaud, and S. Brorson, "Observer bias in randomized clinical trials with measurement scale outcomes: a systematic review of trials with both blinded and nonblinded assessors," *Cmaj*, vol. 185, no. 4, pp. E201–E211, 2013.

[45] R. Goddard and P. Villanova, "Designing surveys and questionnaires for research," *The psychology research handbook: A guide for graduate students and research assistants*, pp. 114–125, 2006.

[46] M. S. Litwin, *How to measure survey reliability and validity*. Sage, 1995, vol. 7.

[47] F. J. Fowler Jr and F. J. Fowler, *Improving survey questions: Design and evaluation*. Sage, 1995.

[48] J. R. Evans and A. Mathur, "The value of online surveys," *Internet research*, vol. 15, no. 2, pp. 195–219, 2005.

[49] K. Poels, Y. A. de Kort, and W. A. IJsselsteijn, "D3. 3: Game experience questionnaire: development of a self-report measure to assess the psychological impact of digital games," 2007.

[50] J. D. Winkler, D. E. Kanouse, and J. E. Ware, "Controlling for acquiescence response set in scale development." *Journal of Applied Psychology*, vol. 67, no. 5, p. 555, 1982.

[51] "Bcs code of conduct," 2019. [Online]. Available: https://www.bcs.org/membership/become-a-member/bcs-code-of-conduct/

[52] "Guide to the general data protection regulation," 2018. [Online]. Available: https://www.gov.uk/government/publications/guide-to-the-general-data-protection-regulation

[53] P. Isaias and T. Issa, "Information system development life cycle models," in *High Level Models and Methodologies for Information Systems*. Springer, 2015, pp. 21–40.

[54] T. Fullerton, C. Swain, and S. Hoffman, *Game design workshop: Designing, prototyping, & playtesting games*. CRC Press, 2004.

APPENDIX A
REFLECTIVE REPORT

The following subsections will discuss the various issues I have encountered during the development of this dissertation. Each issue will declare a SMART goal that can be actioned upon to help better myself in those areas.

A. Artefact Verification

Learning of the need to create unit tests and other verification methods came later into the project than I would have liked. This meant that the unit tests were created after the functions were created. Ideally, according to the values of Test Driven Development (TDD) in particular, unit tests should be written before the functional code is written. This meant that my code had to be refactored slightly to work with the unit tests. Also it meant that not all of the code in the project has unit tests attached. This meant that although regularly smoke tested and verified through other testing means, my project was not as thoroughly robust as it could have been. To remedy this, I will endeavour to learn TDD and attempt to utilize it in a solo project. I have found a collection of TDD exercises at https://github.com/mwhelan/Katas that I can complete in order to learn how to write good unit tests. I will use each exercise as a Kata, a form of deliberate practice, and will aim to complete the first exercise at least ten times until it becomes second nature. I will do this for ten of the exercises, one exercise each week or until I feel confident with that exercise.

B. Approaching Participants

When it came to running the actual experiment, I found myself very anxious about interacting with people I did not know. As an introvert, the idea of randomly asking people to come and help me by playing a game they had no prior interest in and filling out a possibly tedious questionnaire multiple times was not appealing. In fact, the only thing that was a comfort was knowing that I had chosen a time when two of my peers were also asking people to take part in their dissertation experiments. This was helpful as I knew that people would already be cognisant of the fact that they might be asked to do playtests. I also felt slightly awkward and unsure whether I should leave them to carry out the test without me so that I could find more participants or whether I should stay and awkwardly sit a few spaces away in case they had any questions. I chose to do the latter. If I were to carry out experiments like this again in the future, I would want to prepare better beforehand. Therefore, next time I intend to write a short script that includes all the relevant information as well as any likely questions possible participants may ask. I will read and process this information for 15 minutes beforehand. This method also helps when doing presentations which may be a necessity were I to work in the games industry.

C. Planning

During development of the Artefact, several parts of the main game had to be overhauled. This included the AI Director and the Dynamic Difficulty Algorithm itself. This meant that weeks worth of work had to be scrapped because the system simply did not work as intended during testing. This may have been preventable had I developed a suitable Unified Modelling Language (UML) diagram beforehand as they can keep development productive and focused. Learning how to properly create UML diagrams would help to plan out new features before any programming takes place and navigate the source code more easily. Lucidchart is a cloud-based diagramming solution and would allow for the creation of UML diagrams that can evolve with a development project. I will therefore follow the tutorials on https://www.lucidchart.com/blog/types-of-UML-diagrams and learn to create a Class diagram that depicts my Artefact as it currently stands. I will spend one hour a day for one week on this task. I will test the effectiveness of my diagram by giving it to a fellow programmer and asking them: how does the AI Director interact with the enemies and spawners? If they can accurately answer this by looking at my UML diagram alone then I will know I have successfully learnt how to create useful UML diagrams. I will then attempt to use this method before I start coding on my next software project.

D. Level Design

I believe one aspect of the main game that could have been improved was the level design. Although suitable enough for the experiments needs, if the level had been better designed there may have been different results in terms of player experience.

E. Time Management

F. Motivation

G. Random Sampling

While I believe that the results gathered via this study are useful, I believe that a study that had more participants randomly sampled from the real world rather than the Games Studio would produce more useful data. Sampling by convenience from the studio had many benefits. Firstly, it allowed for an easy way to gather the data. Doing an online study or a study that required relocating to an area with normal unbiased participants with no background in game development would have been preferable. However, due to time constraints it was decided that sampling by convenience would be a necessary move. Were I to run an experiment like this in the future, I would prefer not to use a single population group such as games students. However, a way I could have improved would have been to gather the emails of games academy students and randomly select participants from that list. This would have been better than inviting students by convenience.

APPENDIX B
ADDITIONAL MATERIALS

Link to Repository: https://gamesgit.falmouth.ac.uk/scm/~ac200905/dda_artefact.git

Link to AI Director Unit Tests: https://gamesgit.falmouth.ac.uk/users/ac200905/repos/dda_artefact/browse/Assets/Editor/AIDirector_Tests.cs

Link to Player AI testing scene with variables used in testing the difficulty systems: https://gamesgit.falmouth.ac.uk/users/ac200905/repos/dda_artefact/browse/Assets/Scenes/AITestScene.unity

Link to Pilot test and Player AI testing results: https://gamesgit.falmouth.ac.uk/users/ac200905/repos/dda_artefact/browse/Assets/Report

The specific script I have made for the Dynamic Difficulty Algorithm is AIDirector.cs but the entirety of the Unity Project was created and designed for the experiment.

A. *Unit Tests*

Figure 8 shows the unit tests created for the AI Director using the Unity Test Runner.

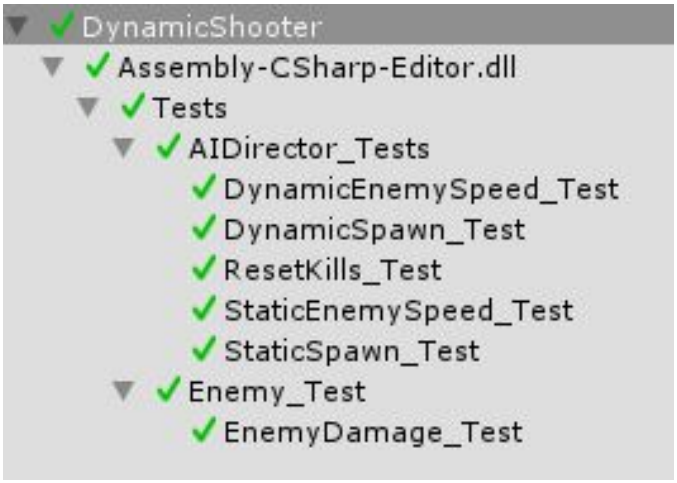

Fig. 8. Passed Unit Tests using Unity's Test Runner

Figure 9 shows the final iteration of the consent form. While originally it was designed to be in paper form, it was decided that implementing it into the beginning of the artefact itself would be more efficient and enables the participant to consent before even being able to access the game.

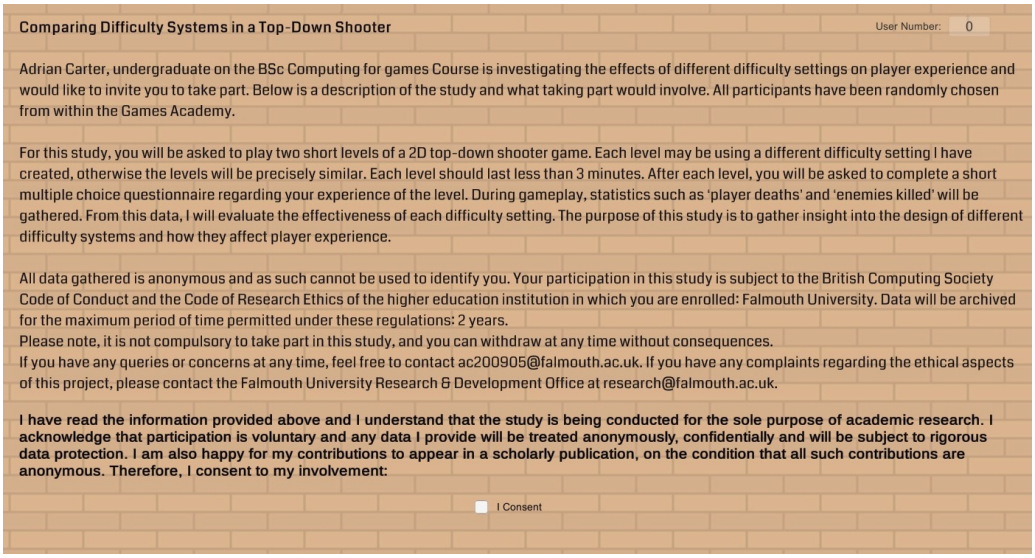

Fig. 9. Consent Form

TABLE III
ARTEFACT USER STORIES

User Story
As an experimenter, I want the dynamic level to adapt to the players skill level
As an experimenter, I want the static level to steadily increase in difficulty
As a player, I want to play a tutorial that introduces the basic mechanics
As a player, I want to be able to move around the level
As a player, I want enemies to spawn out of sight that move into the level and attack me
As a player, I want to be able to shoot and kill enemies
As a player, I want to be able to aim down the sights to increase the accuracy of my shots
As a player, I want to be able to reload my weapon
As a player, I want a HUD that tells me how many bullets I have left in the clip and how much play time is left
As a player, I want to answer a questionnaire after each level
As an experimenter, I want the artefact to record the questionnaire results
As an experimenter, I want the artefact to record the necessary telemetry data
As an experimenter, I want the artefact to output the data to a .csv file on a secure network drive when the game ends

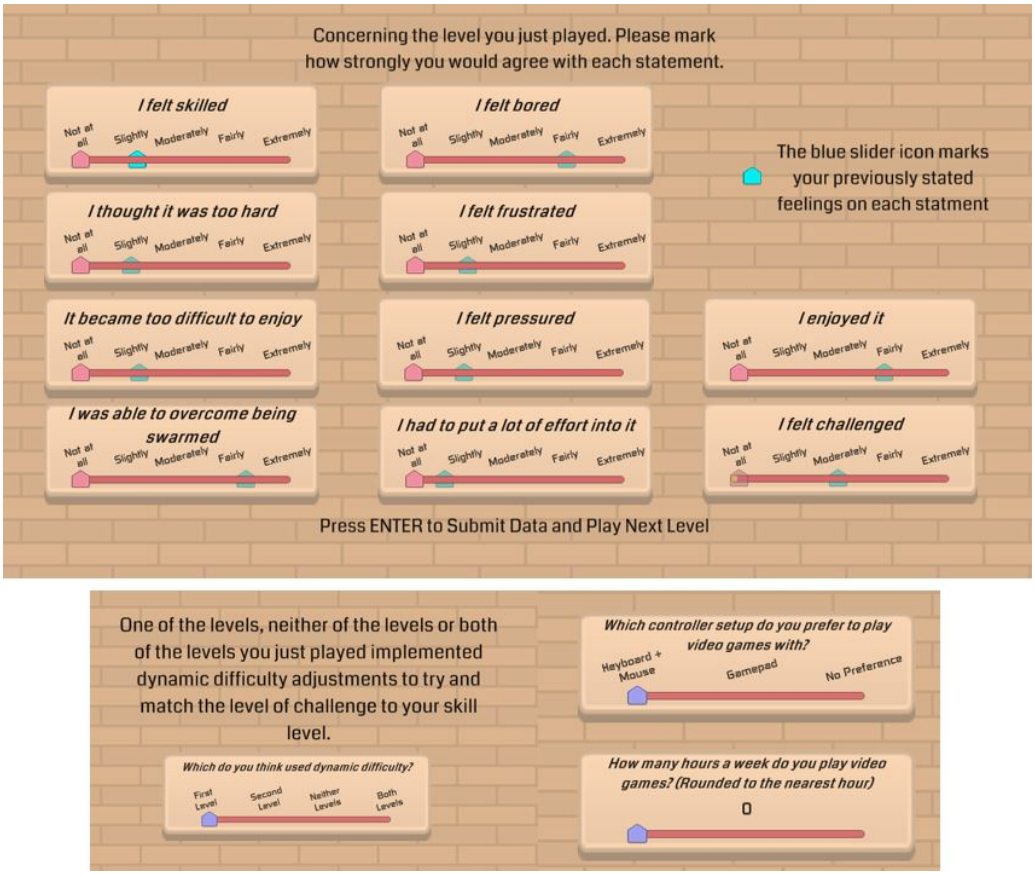

Fig. 10. In-Game Questionnaire

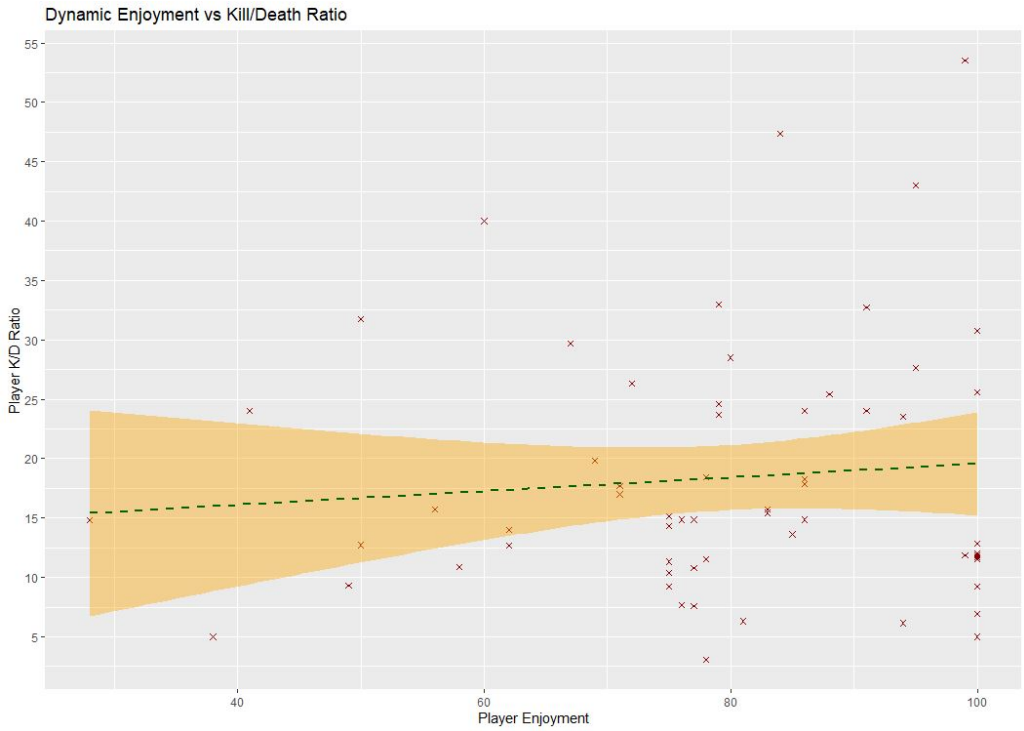

Fig. 11. Scatter plot of Player Enjoyment vs Player Kill/Death Ratio

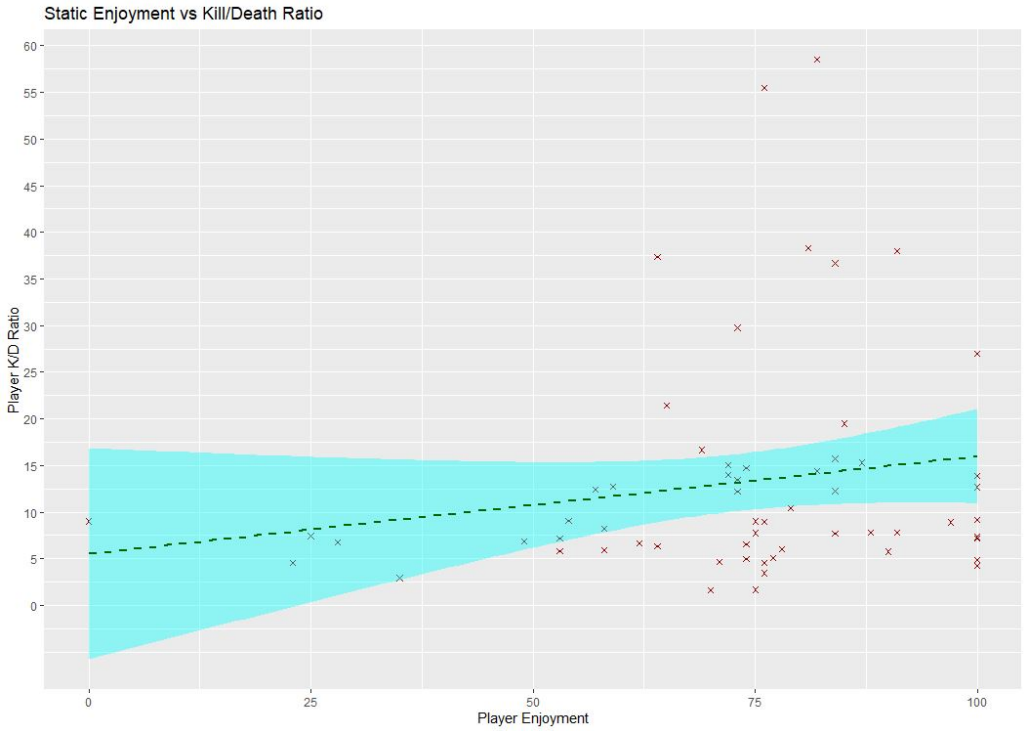

Fig. 12. Static Scatter plot of Player Enjoyment vs Player Kill/Death Ratio

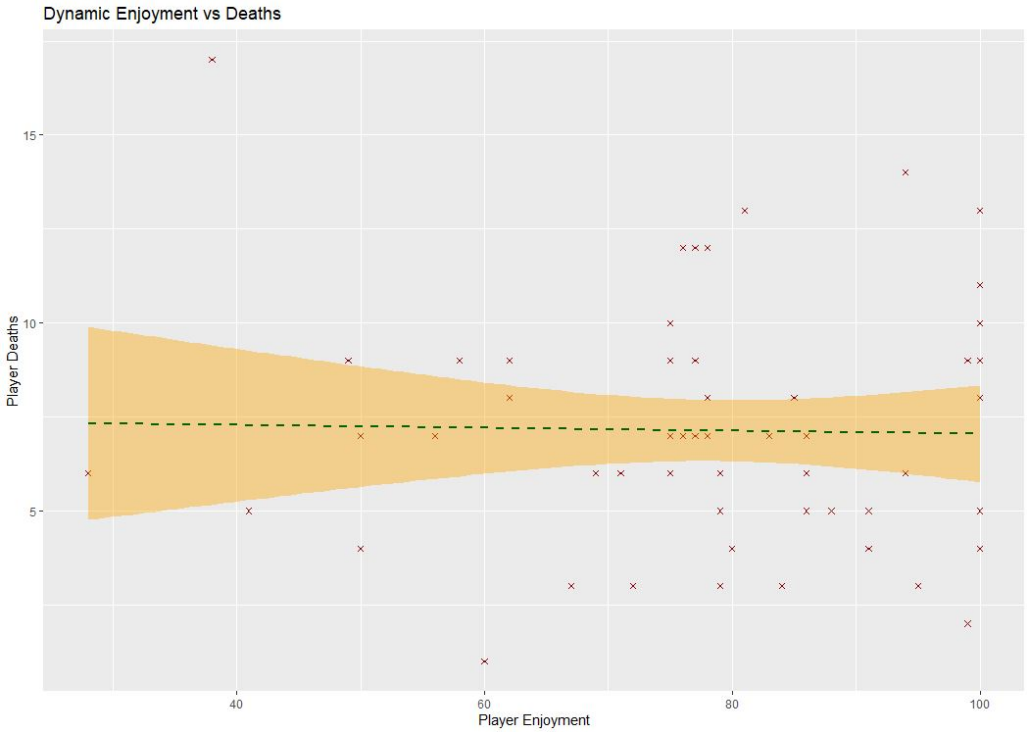

Fig. 13. Dynamic Scatter plot of Player Enjoyment vs Player Deaths

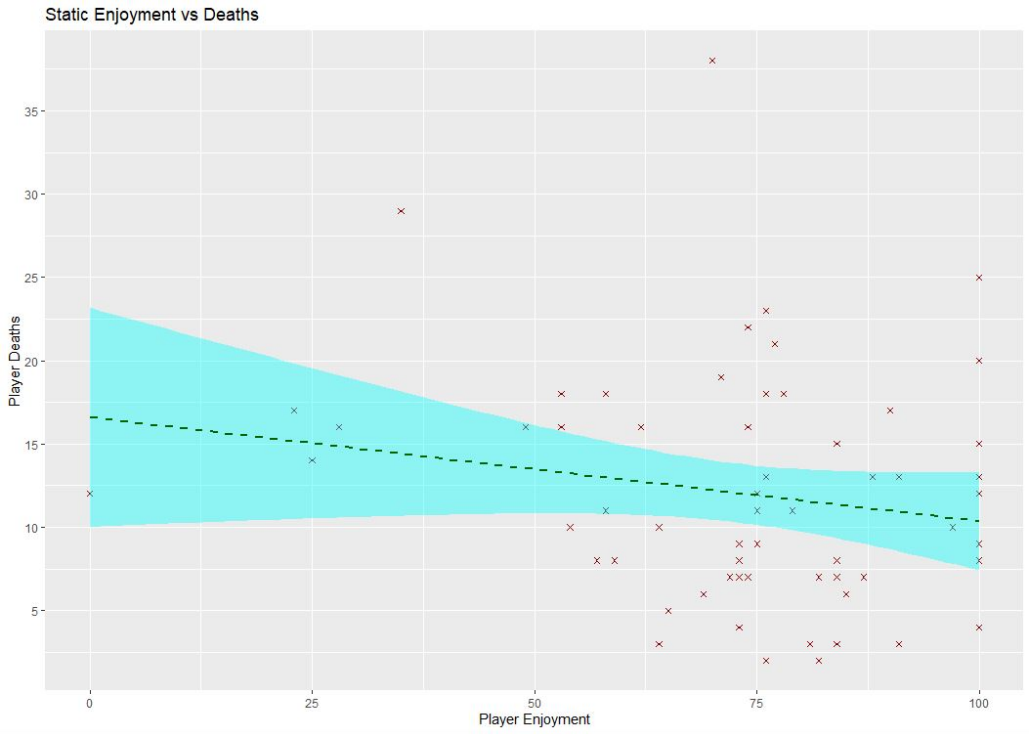

Fig. 14. Static Scatter plot of Player Enjoyment vs Player Deaths

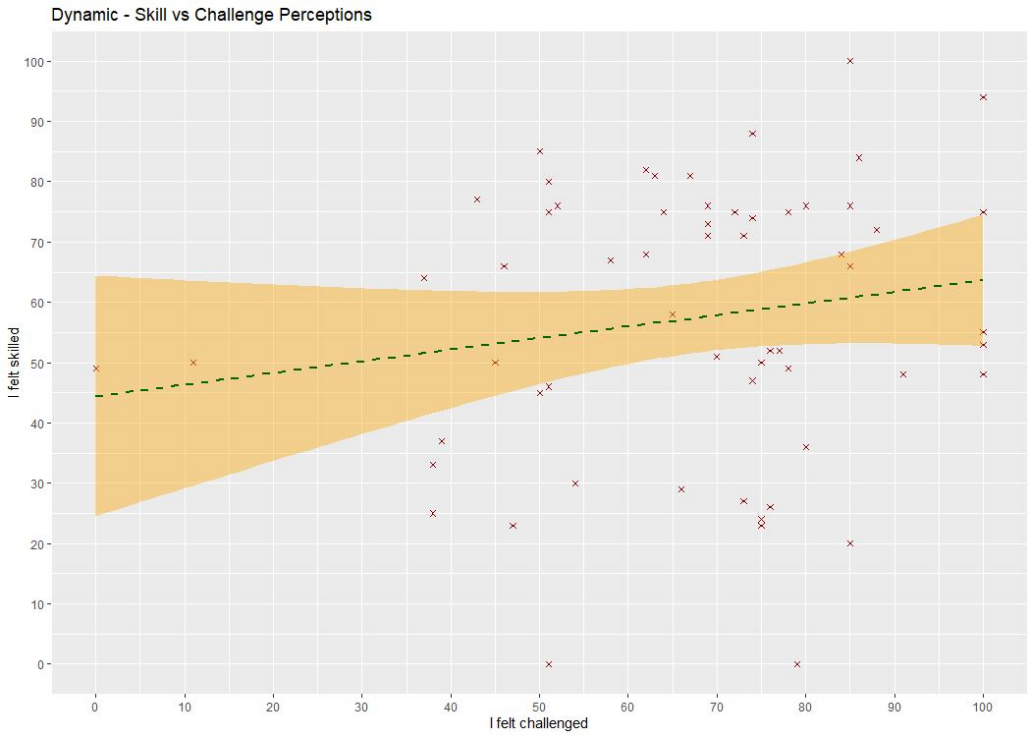

Fig. 15. Dynamic Scatter plot of Skill and Challenge Perceptions

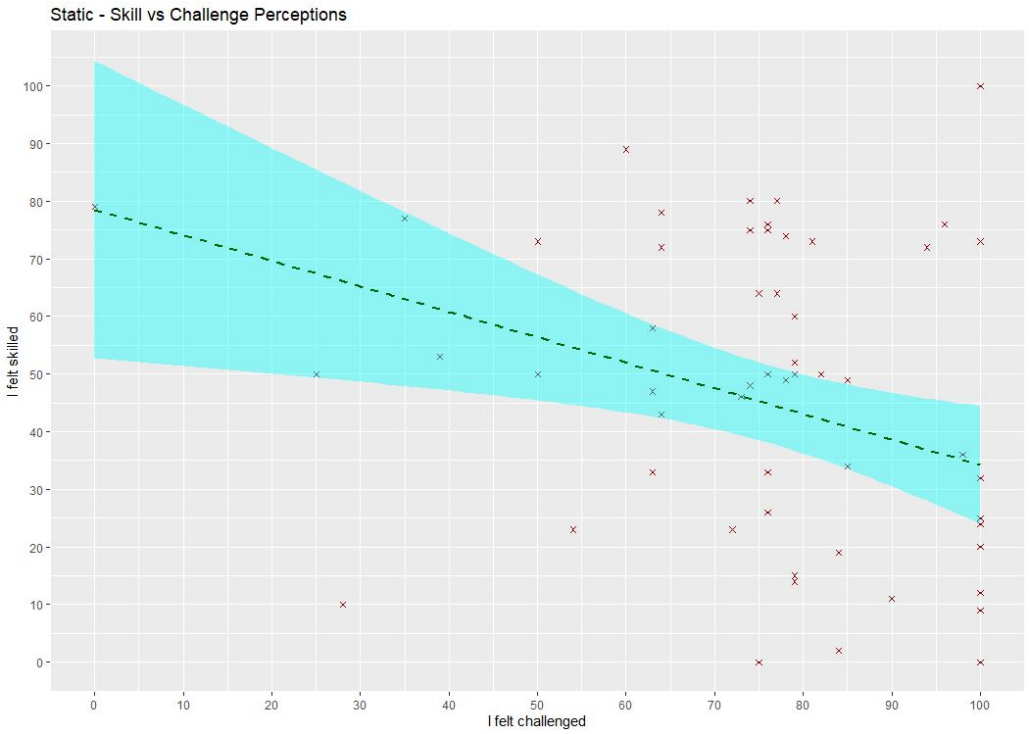

Fig. 16. Static Scatter plot of Skill and Challenge Perceptions

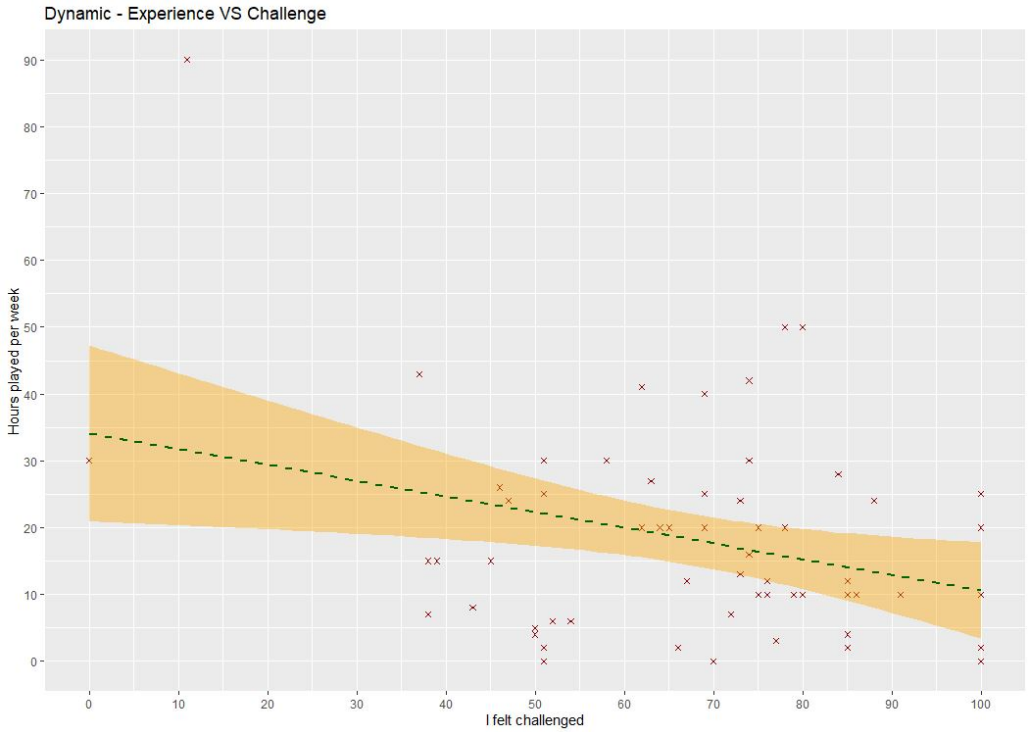

Fig. 17. Dynamic Scatter plot of Experience and Challenge Perceptions

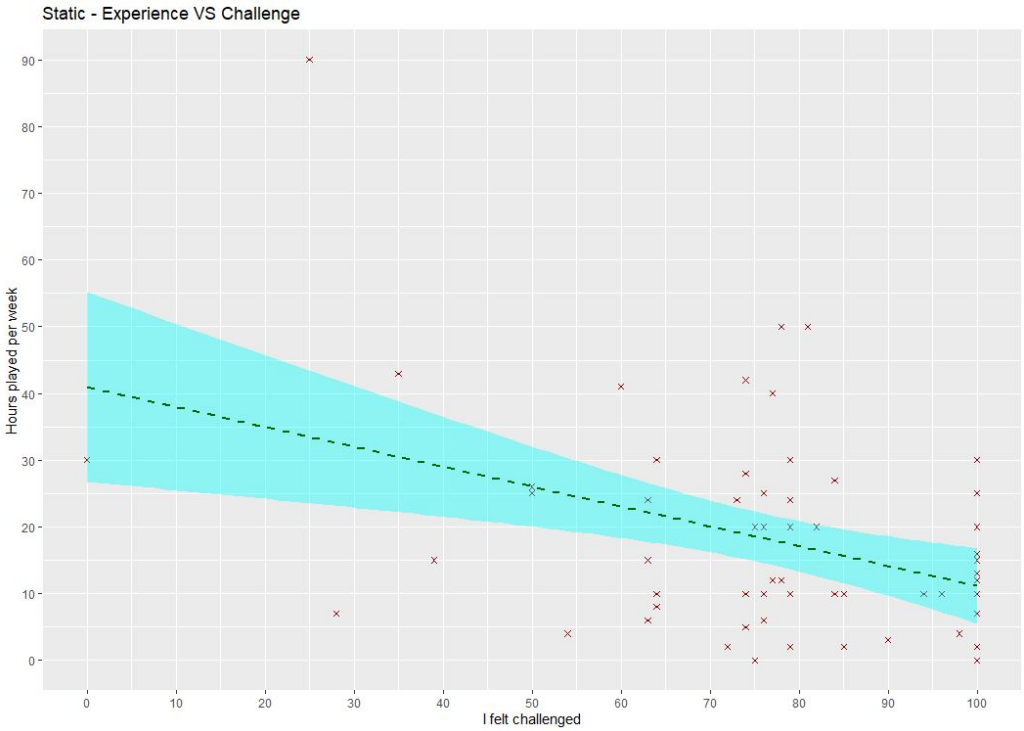

Fig. 18. Static Scatter plot of Experience and Challenge Perceptions

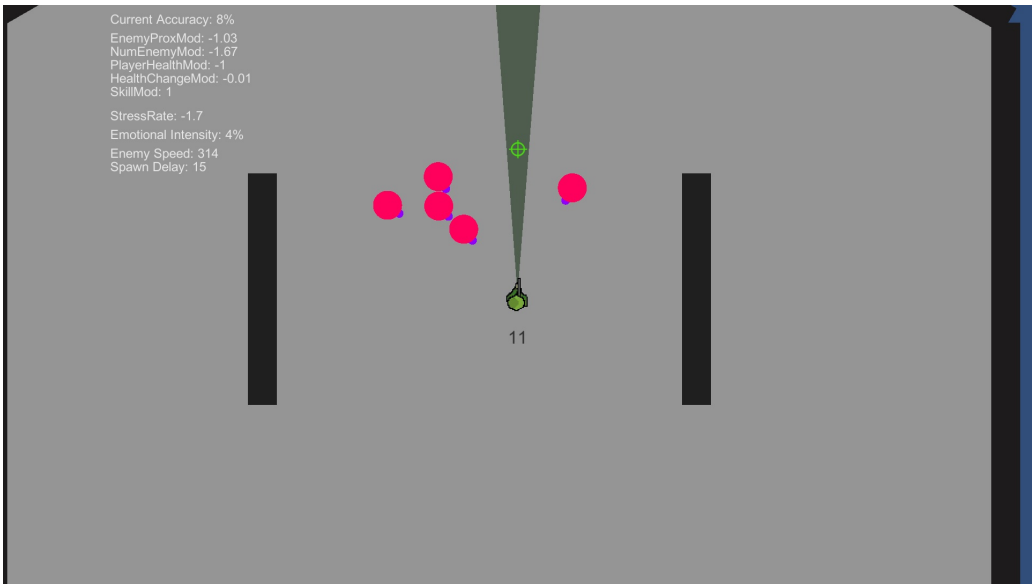

Fig. 19. Initial Bespoke Game Prototype

Fig. 20. Screenshot of the Main Game Level

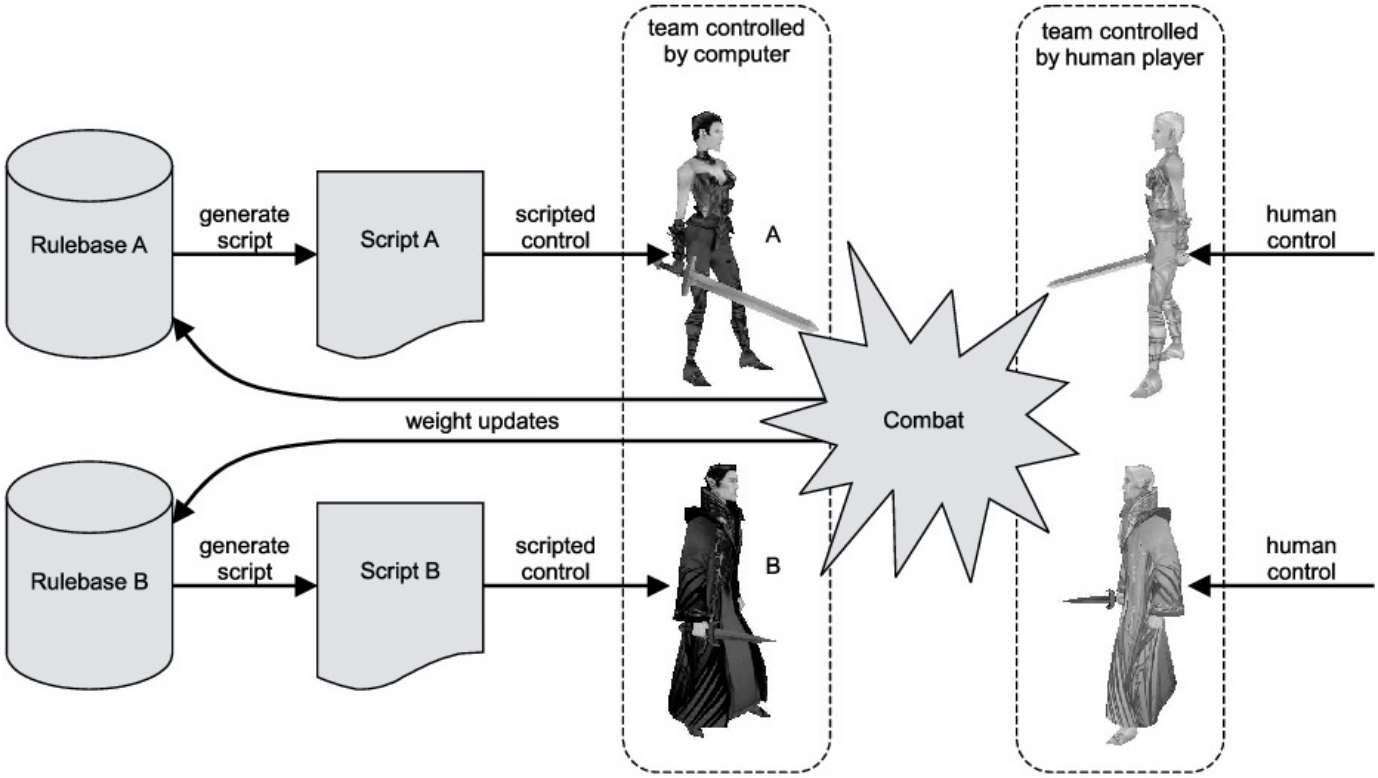

Fig. 21. Dynamic Scripting

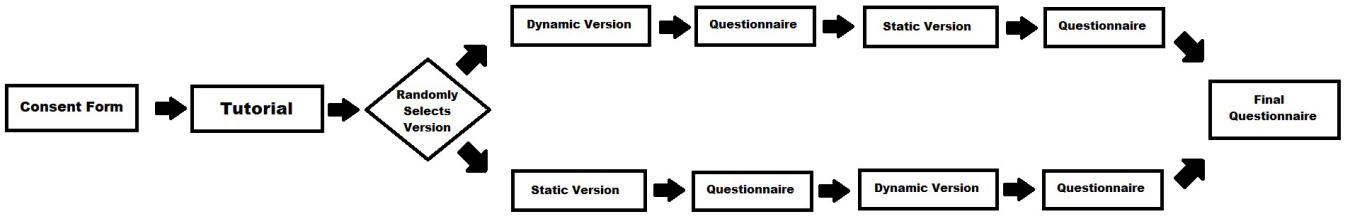

Fig. 22. Experiment Flowchart

```
UpdateSkillGrade
    # Skill rating based on kills
    skillGrade = numEnemiesKilled

    # Half skill rating if player dies
    if (playerJustDied)
        skillGrade = skillGrade / 2
        numEnemiesKilled = numEnemiesKilled / 2

UpdateDynamicSpawnDelay
    # Delay decreases with higher skill rating
    # Delay increases with lower skill rating
    spawnDelay = maxSpawnDelay - (skillGrade * modifier)

UpdateDynamicEnemySpeed
    # Speed increases with higher skill rating
    # Speed decreases with lower skill rating
    enemySpeed = minEnemySpeed + (skillGrade * modifier)
```

Fig. 23. The Dynamic Algorithm

```
UpdateStaticSpawnDelay
    delayDiff = maxSpawnDelay - minSpawnDelay
    delayIncreasePerSec = delayDiff / totalLevelTime
    spawnDelay = currentSpawnDelay - (delayIncreasePerSec * deltaTime)

UpdateStaticEnemySpeed
    speedDiff = maxEnemySpeed - minEnemySpeed
    speedIncreasePerSec = speedDiff / totalLevelTime;
    enemySpeed = enemySpeed + (speedIncreasePerSec * deltaTime)
```

Fig. 24. The Static Algorithm

```
#install.packages("readr")
#install.packages("psych")

library(readr)
library(ggplot2)

# Find CSV data set and import it
dat <-read.csv("D:/user_data.csv")

# T-Test for Hypothesis 1 -----------
# Testing to see if players enjoyed the dynamic version more than the static
funTest <- t.test(dat$FunScoresDynamic,dat$FunScoresStatic,paired=TRUE, alternative="greater")

# If not normally distributed - paired dependent
funTest <- wilcox.test(dat$FunScoresDynamic,dat$FunScoresStatic, paired = TRUE,
alternative="greater")

funTest$p.value
funTest$estimate

# Check if normal distribution
ggplot(data=dat, aes(dat$StaticEnjoyed)) +
+     geom_histogram(breaks=seq(0, 100, by=1))

# F-Test for Hypothesis 2 ------------
# Testing to see if the variances of player challenge scores in static version is higher than
# the variance in player challenge scores in the dynamic version
challengeTest2 <- var.test(dat$ChallengeScoresStatic, dat$ChallengeScoresDynamic, alternative
= "greater")

# F-Test for Hypothesis 3 ------------
# Testing to see if the variances of player deaths in static version is higher than the varian
in player deaths in the dynamic version
challengeTest2 <- var.test(dat$Static.Deaths, dat$Dynamic.Deaths, alternative = "greater")

# Chi-Square Test for Hypothesis 4 ---------
correctVsIncorrect < c(numCorrect, numIncorrect)

# Chi square test to see if players guessed the dynamic version correctly more significantly
# often than by random chance
versionChoiceRandomTest <- chisq.test(correctVsIncorrect, p = c(1/2, 1/2))

# If answers were not given at the rate of random chance, were answers more often correct?
versionChoiceCorrectTes2t <- chisq.test(correctVsIncorrect, p = c(4/5, 1/5))


# For pilot tests ---------
pilotKDTest <- t.test(dat$EnemiesKilledStatic, y, paired = TRUE, alternative = "two.sided")

# Use this scatter graph to see if Kill/Death Ratios are similar between players using the
# dynamic setting
ggplot(dat, aes(x=dat$PlayerKDRatioDynamic, y=dat$PlayerKDRatioDynamic)) + geom_point()
+ geom_smooth(method=lm)
```

```
# Graphs to look for correlations in the data I will be collecting

# Box plot showing the differences in fun between both difficulty settings
boxplot(dat$FunScoresDynamic, dat$FunScoresStatic)

# Box plot showing the differences in challenge between both difficulty settings
boxplot(dat$ChallengeScoresDynamic, dat$ChallengeScoresStatic)

# Fun and Challenge together
boxplot(dat$FunScoresDynamic, dat$FunScoresStatic,

dat$ChallengeScoresDynamic, dat$ChallengeScoresStatic,
names = c("Enjoyed", "Enjoyed",

"Challenged", "Challenged" ),
main = "Player Enjoyment and Challenge", col=(c("orange","cyan")),
ylab = "% Agreement")
legend("top", inset=.07, cex=0.8, legend = c("Dynamic","Static") ,
col = c("orange", "cyan"), fill = c("orange", "cyan"))


# Boxplot with both challenge and fun
boxplot(dat$ChallengeScoresDynamic, dat$ChallengeScoresStatic,
dat$FunScoresDynamic, dat$FunScoresStatic,
names = c("Dynamic Challenge", "Static Challenge", "Dynamic Fun",
"Static Fun"),
main = "Comparing Fun and Challenge", col=(c("orange","cyan")),
ylab = "% Agreement")

# Boredom and frustration comparisons
boxplot(dat$Dynamic...I.felt.bored, dat$Static...I.felt.bored,
dat$Dynamic...I.felt.frustrated, dat$Static...I.felt.frustrated,
names = c("Dynamic Boredom", "Static Boredom", "Dynamic Frustration",
"Static Frustration"),
main = "Comparing Boredom and Frustration", col=(c("orange","cyan")),
ylab = "% Agreement")

# Boxplot for 'Too hard to enjoy' and 'overcame being swarmed'
boxplot(dat$Dynamic...It.became.too.difficult.to.enjoy,
dat$Static...It.became.too.difficult.to.enjoy,
dat$Dynamic...I.was.able.to.overcome.being.swarmed,
dat$Static...I.was.able.to.overcome.being.swarmed,
names = c("Too Hard To Enjoy", "Too Hard To Enjoy",
"Overcame Swarm", "Overcame Swarm" ),
main = "General Difficulty Perceptions", col=(c("orange","cyan")),
ylab = "% Agreement")

legend("top", inset=.07, cex=0.8, legend = c("Dynamic","Static") ,
col = c("orange", "cyan"), fill = c("orange", "cyan"))
```

```
# Boxplot of Questionnaire Results
boxplot(dat$FunScoresDynamic, dat$FunScoresStatic,
dat$ChallengeScoresDynamic, dat$ChallengeScoresStatic,
dat$Dynamic...I.thought.it.was.too.hard,
dat$Static...I.thought.it.was.too.hard,
dat$Dynamic...I.felt.skilled, dat$Static...I.felt.skilled,
dat$Dynamic...I.felt.bored, dat$Static...I.felt.bored,
dat$Dynamic...I.felt.pressured, dat$Static...I.felt.pressured,
dat$Dynamic...I.felt.frustrated, dat$Static...I.felt.frustrated,
names = c("Enjoyed", "Enjoyed",
"Challenged", "Challenged",
"Too Hard", "Too Hard",
"Skilled", "Skilled",
"Bored", "Bored",
"Pressured", "Pressured",
"Frustrated", "Frustrated"),
main = "Questionnaire Perception Results", col=(c("orange","cyan")),
ylab = "% Agreement")

legend("top", inset=.07, cex=0.8, legend = c("Dynamic","Static"),
col = c("orange", "cyan"), fill = c("orange", "cyan"))

# Examples of scatter graph R code ------------------

# Scatter graph with line of best fit to see if the more a player
# dies the less they enjoyed the game
ggplot(dat, aes(x=FunScoresDynamic, y=PlayerDeathsStatic)) + geom_point() +
geom_smooth(method=lm)

# Dynamic - Player Deaths vs enjoyment
ggplot(dat, aes(x=FunScoresDynamic, y=Dynamic.Deaths)) +
geom_point(shape=4, color="darkred") +
geom_smooth(method=lm, linetype="dashed",
color="darkgreen", fill="orange") +
xlab("Player Enjoyment") + ylab("Player Deaths") +
ggtitle("Dynamic Enjoyment vs Deaths") +
scale_y_continuous(breaks = c(0,5,10,15,20,25,30,35,40))

# Static - Player Deaths vs enjoyment
ggplot(dat, aes(x=FunScoresStatic, y=Static.Deaths)) +
geom_point(shape=4, color="darkred") +
geom_smooth(method=lm, linetype="dashed",
            color="darkgreen", fill="cyan") +
xlab("Player Enjoyment") + ylab("Player Deaths") +
 ggtitle("Static Enjoyment vs Deaths") +
scale_y_continuous(breaks = c(0,5,10,15,20,25,30,35,40))
```