Electronics and Computer Science
Faculty of Engineering and Physical Sciences
University of Southampton

Harry Nelson
May 2022
Virtual Mentor for a Competitive Tactical Shooter

Project Supervisor: Dr. David Millard
Second Examiner: Dr. Son Hoang
A project report submitted for the award of MEng Computer Science.

Abstract

In video games players are often provided with a tutorial to teach the basics required to play, and then learn more advanced mechanics as they progress through the game. For competitive video games however, players often need to learn how to play externally due to unknown mechanics or strategies that are not introduced in a tutorial. This project explores the potential for an in-game virtual mentor to help a player train the mechanics of a game by implementing it into an existing FPS prototype.

A basic FPS game was created with mechanics similar to currently popular tactical FPS games, such as CS:GO or Valorant. A learning system was implemented into this that tracked the performance of players and gave them live feedback after each kill as they played. It also made use of Dynamic Difficulty Adjustment to induce a state of flow in players to make learning more effective.

FPS players of different skill levels then played this FPS and responded to questions about their experience with the feedback system. All players improved over the course of the trial, and almost all thought the combination of live feedback and dynamic difficulty was effective for improvement.

Statement of Originality

- I have read and understood the ECS Academic Integrity information and the University's Academic Integrity Guidance for Students.
- I am aware that failure to act in accordance with the Regulations Governing Academic Integrity may lead to the imposition of penalties which, for the most serious cases, may include termination of programme.
- I consent to the University copying and distributing any or all of my work in any form and using third parties (who may be based outside the EU/EEA) to verify whether my work contains plagiarised material, and for quality assurance purposes.

We expect you to acknowledge all sources of information (e.g. ideas, algorithms, data) using citations. You must also put quotation marks around any sections of text that you have copied without paraphrasing. If any figures or tables have been taken or modified from another source, you must explain this in the caption and cite the original source.

I have acknowledged all sources, and identified any content taken from elsewhere.

If you have used any code (e.g. open-source code), reference designs, or similar resources that have been produced by anyone else, you must list them in the box below. In the report, you must explain what was used and how it relates to the work you have done.

- Colanderp's Unity Tutorial Project (`https://github.com/Colanderp/UnityTutorials`)
- Jose Diaz Inverse Kinematics Module (`https://assetstore.unity.com/packages/tools/animation/inverse-kinematics-1829`)
- Kevin Iglesias' 3D Character Dummy (`https://assetstore.unity.com/packages/3d/characters/humanoids/humans/3d-character-dummy-178395`)
- Free textures from `https://freestocktextures.com/texture/white-grunge-wall,1420.html` and `https://unsplash.com/photos/PdGBci-4jR8`
- Free music from `https://www.chosic.com/free-music/all/` and `https://www.purple-planet.com/`

You can consult with module teaching staff/demonstrators, but you should not show anyone else your work (this includes uploading your work to publicly-accessible repositories e.g. Github, unless expressly permitted by the module leader), or help them to do theirs. For individual assignments, we expect you to work on your own. For group assignments, we expect that you work only with your allocated group. You must get permission in writing from the module teaching staff before you seek outside assistance, e.g. a proofreading service, and declare it here.

I did all the work myself, or with my allocated group, and have not helped anyone else.

We expect that you have not fabricated, modified or distorted any data, evidence, references, experimental results, or other material used or presented in the report. You must clearly describe your experiments and how the results were obtained, and include all data, source code and/or designs (either in the report, or submitted as a separate file) so that your results could be reproduced.

The material in the report is genuine, and I have included all my data/code/designs.

We expect that you have not previously submitted any part of this work for another assessment. You must get permission in writing from the module teaching staff before re-using any of your previously submitted work for this assessment.

I have not submitted any part of this work for another assessment.

If your work involved research/studies (including surveys) on human participants, their cells or data, or on animals, you must have been granted ethical approval before the work was carried out, and any experiments must have followed these requirements. You must give details of this in the report, and list the ethical approval reference number(s) in the box below.

ERGO/FEPS/41626.A2

Contents

1 Problem Statement and Goals of Project

1.1 Problem

Video games usually feature a tutorial to give players the minimum knowledge required for playing the game. Having portions of a game dedicated to helping a player learn means they can figure out how to play the game through guided experience instead of spending time reading externally, and preventing unknown mechanics being frustrating and turning players away from the game. In some games learning the mechanics as you play is an intentional design decision, with a player feeling more powerful towards the end of the game having mastered the systems. However, in competitive player vs player video games such as Counter Strike or League of Legends, as the mechanics can be quite complex and full of nuance that even develops over time as people play, there are no real existing mechanisms to support improvement past a beginner level. In these games the intention is you learn by practicing, playing games and figuring out for yourself what works; however, going into a game against players with knowledge that you lack can be quite demoralising, causing problems in retention for new players.

1.2 Goals

This project aims to develop a prototype FPS game with suitable complexity for there to be a range of mechanical abilities for a player to improve upon. A coaching engine would then be implemented into this FPS game to guide a player's experience using statistics and heuristics to decide where the player is struggling based on their performance and habits tracked over playing (e.g. they tend to enter engagements looking in sub-optimal directions, or often miss because they don't stop moving before shooting). This feedback will be presented to the player visually via the UI and will give the player opportunities to practice a specific skill environmentally. Finally it will try and evaluate the effectiveness of this coaching mechanism by getting FPS players of a range of skill levels to trial the system and give feedback on their experience.

1.3 Scope

Due to the range of complex competitive PVP games, this project aims to experiment with these techniques to aid improvement in a specific situation, rather than providing a framework for all competitive PVP games. Because of this it will focus on teaching a set of mechanics specifically from the tactical FPS genre, and will only be a prototype of sufficient complexity for the coaching engine to have a meaningful presence, but not be a fully fledged competitive FPS in itself. It will be made in Unity on top of an existing, open-source FPS prototype with the engine tuned to add the mechanical complexity required, and then with the coaching features implemented on top of that.

2 Background Research

2.1 Literature Review

2.1.1 Introduction

The aim of this literature review is to investigate existing methods for training in games, and also effective learning environments and methods, to try and formulate a plan for the structure of the game. The ”flow state” was found to be a common method for encouraging engagement, mentioned in game blogs and the game design course at the University of Southampton, so research was done into the effectiveness of that and some examples of how it has been measured and utilised in the past. Methods of how to induce a flow state were also investigated, focusing around the idea of Dynamic Difficulty Adjustment to adjust the gameplay to keep the user engaged. Finally different methods of communicating feedback, both in games and in other fields, were investigated and evaluated.

2.1.2 Flow State

The state of flow is the idea of an ”optimal experience” in an activity where a person is ”completely [absorbed] in the present moment”, to allow the person to experience ”intense and focused concentration” as well as several other benefits that maximise the enjoyment of an activity [1, p. 195]. This state is invoked by balancing the ”ratio between perceived challenges and skills” to avoid boredom at the activity being too easy or anxiety at potential difficulty [1, p. 196].

This state of flow has been researched and observed by Csikszentmihalyi et al. across groups of gender, class, and activity and the causes and benefits remain consistent [1, p. 196]. As video games are usually a product made with the intention of earning money, player enjoyment is a goal worth prioritising, as if a player is not enjoying a game, they can stop and play another. Because of this the characteristics of a state of flow are very similar to the state a video game wants a player to achieve, and so games often use flow methodology to maximise this. The most notable example is the game ”fl0w”, which is entirely centered around a mechanic allowing a player to constantly achieve a state of flow by choosing their own difficulty through ascending or descending while playing the game [2].

In video games specifically, it is easy to provide the ”challenging activity that requires skill” and ”clear goals and feedback” that Schreiber writes about in his game design blog necessary for achieving flow [3]. Csikszentmihalyi et al. note the experience from flow has a role in development of a skill due to the heightened engagement and focus allowing for a more productive session, and research has been done into the effect of flow on learning in games specifically [1]. Perttula et al. found a positive correlation between flow and learning in a literature review in 2017, due to the feelings of satisfaction and enjoyment afterward helping players stay motivated and facilitating the learning [4]. However, the ways that previous research has collected data makes it difficult to be generalised, and it calls for more research to be done into the relationship between game mechanics and the flow experience, developing more measures of flow that can be applied more generally, as well as more studies with larger sample sizes. After this, attempts have been made into making frameworks for measuring the experience of challenge in games. Denisova et al. surveyed several thousand players with a questionnaire featuring many strongly agree/strongly disagree scaled questions, which were then cut down based on similarity and sorted into 4 components, allowing them to rate games based on "emotional", "decision-making", "performative" and "cognitive" challenge [5]. Despite being a good method of more objectively defining how challenging a game was, it required a large data set and requires feedback from many players of a game to be able to compare it to the others.

A qualitative study by Sanjamsaiet al. attempted to measure the impact of flow states on players by surveying a sample of university students who had an interest in playing computer games of any genre or platform [6]. It found that experiencing focus and concentration in a flow state while playing could improve recollection, problem-solving and analysis skills while playing, suggesting that the flow state makes players better in the moment as well as providing a suitable environment for training. This is similar to the outcome of a study by Liu et al. into the impact of competitive games on employee motivation and improvement, that found players put more effort into gameplay against equally skilled opponents [7]. This leads to the idea that a player seeking to improve their skill at a competitive game would likely have a more productive session if their training resulted in a state of flow, where they felt challenged but competent.

2.1.3 Inducing a Flow State

Schreiber outlines several methods for influencing difficulty used in games, as to induce a flow state in a game the difficulty for the player must be balanced. [3]. Methods suggested such as dynamic difficulty adjustment or human opponents are much more appropriate for FPS games. For my project DDA will likely be much more useful, allowing a player to play the game and be guided based on their performance. Hunicke's study into DDA in single player FPS games found that even relatively basic adjustments (along the lines of slightly reducing health points of enemies or increasing supply of health items) were mostly imperceptible to the player while improving performance and without detracting from the players' "sense of agency and accomplishment" [8].

In general there are many methods for implementing DDA in games. A review of different methods by Zohaib summarised that DDA should "track player ability and rapidly adapt", "follow the player's improving or falling level", and "not be clearly perceived by the players" [9]. Some methods they have classified include a "Hamlet System", where difficulty is based on the state of the players inventory, with the theory that they will be using more items such as health potions if they are finding the game more difficult, and so the difficulty should be lowered, and vice versa. Another example is "Dynamic Scripting" where the rules for AI opponents to respond to are adjusted after each encounter with the player in order to generate new strategies, or a reinforcement learning approach that similarly adapts the opponent AI to the players strategies in order to provide engaging encounters using more in depth ML techniques. These may be worth considering, as an AI that can adapt and ramp up difficulty for players as they get better would be ideal for teaching, as long as the AI was an accurate representation of human players. Examples of some DDA strategies used in modern multiplayer games include the racing game "Mario Kart" basing the items players receive on their distance from the player in 1st place, so those falling behind are provided more of a chance to catch up, or the "AI Director" from "Left 4 Dead", a co-op survival FPS that generates situations for the players as they progress through the game to keep players engaged and create fresh situations [10] [11].

DDA for improvement specifically has also been researched. A study in 2018 by Demediuk et al. attempted to measure a player's skill in the fighting game FightingICE (similar to Street Fighter) [12]. They designed an Outcome-Sensitive Action Selection (ROSAS) bot for players to play against that would build a search tree similar to a Monte Carlo Tree Search of "playouts" of the game and choose the tree closest to a drawn game. The bot would then adapt to the players' skill based on the magnitude from which they deviated from the chosen path. The bot adapted to players' perceived skill quite well; however, they note they were unable to obtain proper external ratings of player skill to compare to. This is an interesting method of developing an adversary for the purposes of training in a competitive context, but due to the limited controls and playstates of a fighting game it may prove much more feasible for games of that genre than for a 3D FPS.

2.1.4 Giving Feedback

One of the other main aspects of inducing flow in video games Schrieber points out is providing "clear goals and feedback" to allow them to know how to improve [3]. A masters thesis by Atanasov on designing games around feedback had an analysis of some effective feedback methods in previous games, such as visual effects to notify a player of successful hits in Street Fighter IV, feedback via peripherals through haptic feedback for example to accompany a crash or an area that can't be passed through, or mechanics similar to DDA that adapts behaviour of opponents to player actions in a way they can notice instinctively and force them to change their strategy [13]. While this does give a good idea of the influence and presence of feedback in games, this is quite a general overview on feedback of all kinds, not just specifically for learning. Some research into real-world feedback methods for learning by Jug et al. mentions several techniques to maximise effectiveness, such as providing "direct observations" of issues for the receiver to analyse and learn from, as well "offering suggestions for improvement" so the receiver has an idea of where to go, as opposed to simply relaying statistics for example [14].

Similar suggestions are reported in a video game specific context by Madigan, noting that video games are mostly comprised of fairly simplistic feedback that fails to take human strategies (the ”meta” of a game) into account, for example a competitive game that simply gives you your Kill/Death ratio at the end of a game [15]. The blog then mentions a player-created tool called ”WoWAnalyzer” that creates much higher detail feedback based on an analysis of a game, giving feedback to users using methods very similar to those that the paper by Jug et al. outlined [16] [14]. The player is given a specific situation in which they performed sub-optimally, a forward-thinking suggestion as to what could be done to improve it, and an explanation about why it is important, in order to make it easier for the player to understand the need and the method [17].

2.2 Similar Systems and Approaches

Some popular competitive FPS games have training tools implemented within or externally, due to accessible APIs or support for community additions. Counter-Strike: Global Offensive has a community workshop of maps for users to upload or download from. For example, allowing for user-created maps like the YPRAC series that allow players to choose and customise drill-like practice sessions on maps from the main game [18] [19]. Some features of these custom practice maps, such as general survival modes, use some flow techniques mentioned before to make the gameplay similar to the game it is emulating rather than just purely practicing your aim, in order to help the knowledge stay in the players' minds when they move to a real match. Another tool for CS:GO is Leetify, which analyses your competitive games based on the available recordings and tells you areas you are struggling with in that specific game as well as in general (such as average distance crosshair moved per kill, or positioning), compared to other players of a similar level [20].

Rainbow 6 Siege, another competitive FPS game, has a survival mode where a player can play against a team of bots of their chosen difficulty on a map of their choice, in order to practice aim or strategy; however, the bots they play against do not change their behaviour on higher difficulties [21]. The number of enemies and time to complete the objective are all that is adjusted. Valorant, a game very similar to CS:GO, doesn't have any tools of the same kind as CS:GO for improvement because of the lack of community map support or the ability to get recordings of the games in an accessible way [22].

Generic ”Aim Trainer” style games exist as well, such as Aim Lab or Kovaaks, which provide abstract games where they can analyse and give gameplay tips (e.g. mentioning low reaction time, need to improve tracking ability) [23], [24]. Despite these external tools, there is not much that has been officially designed dedicated to improvement past the initial tutorials.

The key difference between the higher level training or statistics tracking tools and my project is the in-game aspect of tracking a player's weaknesses and presenting solutions to them. It is easy to do drills in external games or practice maps without really understanding what about them you need to improve on and how they apply to a real game, or to see a statistical analysis of your performance but not understand how to apply that knowledge and improve on it when in a real match.

2.3 Conclusion

This is a brief overview of literature covering methods of fostering improvement in video games that could potentially be implemented in this project. There has not been much research done into effective feedback in video games specifically but there are notable examples of projects people have created to bridge the gap between a beginner and a high-rank player, and the literature about feedback for learning in general is also still applicable. Having the player be in a state of flow will maximise enjoyment and also effectiveness of the training, which can be achieved with a form of DDA.

3 Planning and Design

3.1 Solution to the Problem

A system that attempts to foster mechanical improvement should be created. It should give the player prompt feedback on flaws in their mechanical skill per-enemy in a given run of the game, to take advantage of the improved knowledge retention from a state of flow. It also takes advantage of the environment feeling closer to that of the competitive game it is training you for, allowing for the skills to feel applicable outside of the trainer. A state of flow will be created by dynamically adjusting the difficulty of the game based on the player's skill, through changing of the strength of the enemies.

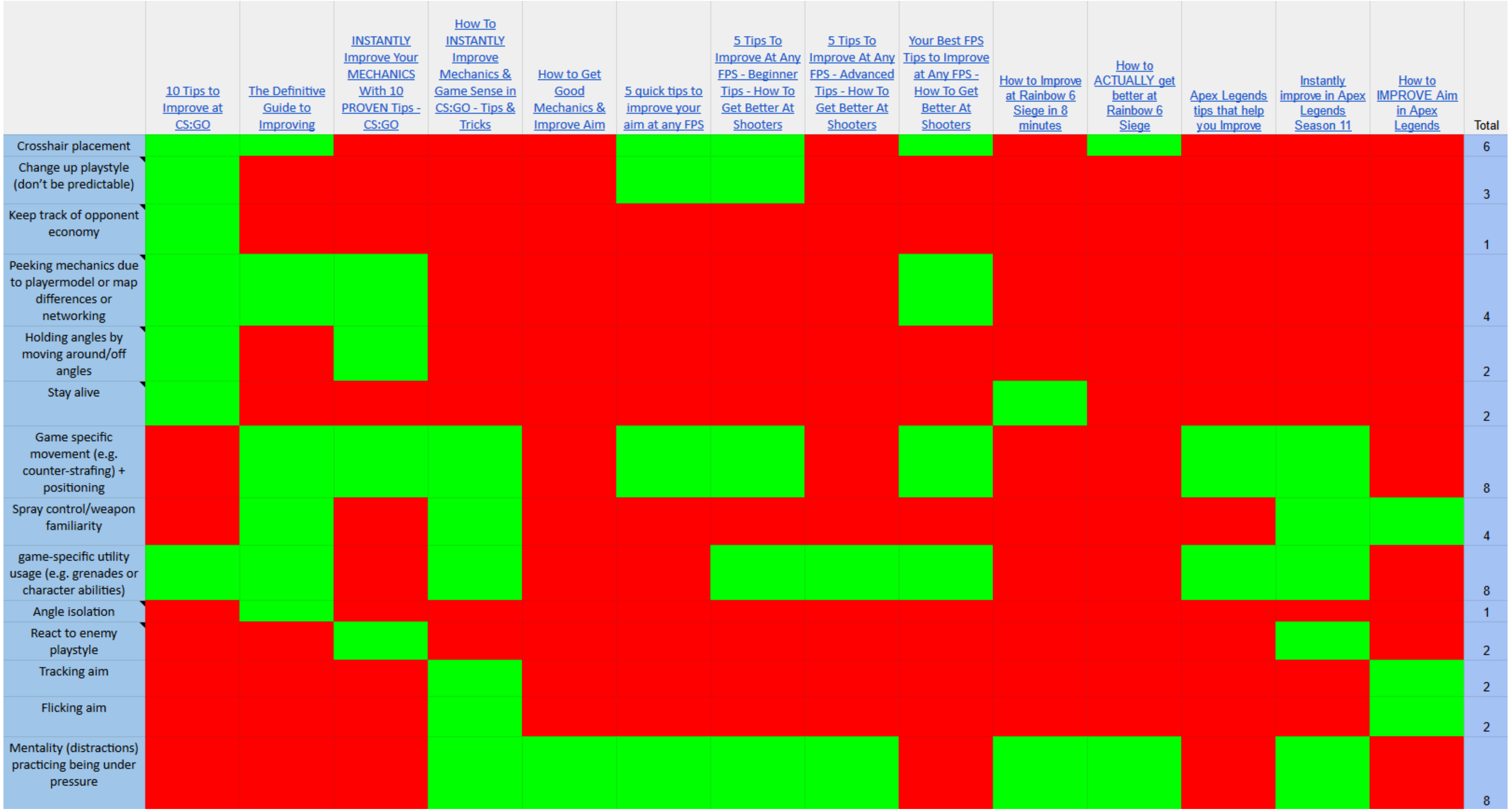
3.2 Requirements Research

3.2.1 Research

To identify the mechanics that should be prioritised, online research was performed for improvement tips from prominent YouTubers, that cross-referenced several videos across different FPS games to focus on higher levels of training that the semi-professional community has outlined as important. This was to select different mechanics and areas for the prototype to focus on improving.

Videos were found by searching for the name of different FPS games (CS:GO, Rainbow 6 Siege, Apex Legends and also "FPS"), selected based on experience with them, and the word "improvement" on YouTube. 14 Videos from YouTubers with a range of followers and view counts were selected, with videos having a minimum of 10,000 views and averaging around 100,000 views, with a positive reception (like/dislike ratio of 95%, analysis was performed before YouTube removed dislike counts [25]). These were then watched and their feedback generalised and compared, allowing each mechanic to be assigned a score based on how frequently they were mentioned. The chart comparing different mechanics can be seen in Figure 1 and all videos watched can be found in the appendix A.

This analysis found that mentality, game-specific utility, and positioning were the most commonly mentioned mechanics, with crosshair placement placed second to those. Aim-based mechanics were less prevalent than expected, however due to the nature of YouTube being a for-profit platform, videos may have given lesser-known advice to distinguish their videos from others available. The table was also sparse, as only 8 or less out of 14 videos agreed on a given mechanic. It was decided due to this to perform a survey on players of an intermediate level to see what mechanics they viewed as the most important.



	10 Tips to Improve at CS:GO	The Definitive Guide to Improving	INSTANTLY Improve Your MECHANICS With 10 PROVEN Tips - CS:GO	How To INSTANTLY Improve Mechanics & Game Sense in CS:GO - Tips & Tricks	How to Get Good Mechanics & Improve Aim	5 quick tips to improve your aim at any FPS	5 Tips To Improve At Any FPS - Beginner Tips - How To Get Better At Shooters	5 Tips To Improve At Any FPS - Advanced Tips - How To Get Better At Shooters	Your Best FPS Tips to Improve at Any FPS - How To Get Better At Shooters	How to Improve at Rainbow 6 Siege in 8 minutes	How to ACTUALLY get better at Rainbow 6 Siege	Apex Legends tips that help you Improve	Instantly improve in Apex Legends Season 11	How to IMPROVE Aim in Apex Legends	Total
Crosshair placement	🟩	🟩	🟥	🟥	🟥	🟩	🟩	🟥	🟩	🟥	🟩	🟥	🟥	🟥	6
Change up playstyle (don't be predictable)	🟩	🟥	🟥	🟥	🟥	🟩	🟩	🟥	🟥	🟥	🟥	🟥	🟥	🟥	3
Keep track of opponent economy	🟩	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	1
Peeking mechanics due to playermodel or map differences or networking	🟩	🟩	🟩	🟥	🟥	🟥	🟥	🟥	🟩	🟥	🟥	🟥	🟥	🟥	4
Holding angles by moving around/off angles	🟩	🟥	🟩	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	2
Stay alive	🟩	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟩	🟥	🟥	🟥	🟥	2
Game specific movement (e.g. counter-strafing) + positioning	🟥	🟩	🟩	🟩	🟥	🟩	🟩	🟥	🟩	🟥	🟥	🟩	🟩	🟥	8
Spray control/weapon familiarity	🟥	🟩	🟥	🟩	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟩	🟩	4
game-specific utility usage (e.g. grenades or character abilities)	🟩	🟩	🟥	🟩	🟥	🟥	🟩	🟩	🟩	🟥	🟥	🟩	🟩	🟥	8
Angle isolation	🟥	🟩	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	1
React to enemy playstyle	🟥	🟥	🟩	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟩	🟥	2
Tracking aim	🟥	🟥	🟥	🟩	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟩	2
Flicking aim	🟥	🟥	🟥	🟩	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟩	2
Mentality (distractions) practicing being under pressure	🟥	🟥	🟥	🟩	🟩	🟩	🟩	🟩	🟥	🟩	🟩	🟥	🟩	🟥	8

Figure 1: Table cross-referencing YouTube Videos on Improvement in FPS games. Green identifies a mechanic being mentioned.

3.2.2 Survey

To get a more specific insight into players' thoughts on mechanics, a survey was presented to students (following the guidelines of ethical approval ERGO/FEPS/41626.A2) using Microsoft Forms (questions in Appendix A). Students from the Video Game society were presented the survey via social media. This survey asked what mechanics were integral to their improvement from beginner to the rank they had obtained now, and also to rank a filtered list of mechanics found from the YouTube Research.

The survey got 45 responses, with 33 of those identifying as intermediate or higher ranked in their respective FPS game. The feedback was generalised to count specific mechanics mentioned and of those the most commonly mentioned were: Aim, Gamesense, Positioning, Crosshair Placement, and Utility Use, which were mentioned 10+ times. All others were mentioned 6 or fewer (see Figure 2).

In the ranked question, the average rankings of each mechanic were taken. The most commonly ranked in the top 3/8 positions were Aim, Crosshair Placement, and Positioning. This contrasted with the research from YouTube, but this is likely because ”Aim”-related mechanics were grouped for the ranking feedback into an ”Aim” category, whereas in the YouTube videos more specific elements like ”spray control”, ”tracking aim” and ”flicking aim” were their own elements (see Figure 3).

Mechanic Mentioned	Number of Occurrences	Explanation
Aim	13	Raw aiming ability, such as ”tracking” or ”flicking”
Gamesense	13	Knowing what decisions to make and when, such as when to remain quiet or when to take a risk
Positioning	12	Staying in cover, not exposing yourself to too many angles, etc.
Crosshair Placement	11	Lining up your crosshair with where you expect a player's head to be to minimise the time to damage
Game Specific Utility	10	Grenades in CS:GO, abilities in Overwatch, and other mechanics that complement the shooting gameplay but aren't the primary focus
Game Knowledge	6	Environmental knowledge such as map layouts or ability cooldowns etc.
Movement	5	Things like counter-strafing in CS:GO, using knowledge of the game's movement system to make yourself harder to hit or make it easier to hit enemies
Communication	5	Ability to efficiently communicate knowledge with your team as you learn it across the game
Teamwork	4	Working with your team to execute strategies
Friendly/Enemy Economy Tracking	4	For games where it is necessary, knowing when your team or the enemies have access to specific weapons or abilities so they can be strategised around
Spray Control / Weapon Familiarity	3	Knowing the specific pattern or behaviour of a weapon so that you can use it effectively.

Figure 2: Sum of the number of mentions in free text question from mechanics survey.

Response	Average Placement (Index 0)	Explanation
”Aim” In General	1.22	Aim abilities such as ”tracking” or ”flicking”
Crosshair Placement	1.78	Lining up your crosshair with where you expect a player’s head to be to minimise the time to damage
Positioning	1.91	Staying in cover, not exposing yourself to too many angles, etc.
Game Specific Utility	2.78	Grenades in CS:GO, abilities in Overwatch, and other mechanics that complement the shooting gameplay but aren’t the primary focus
Spray Control / Weapon Familiarity	2.96	Learning the behaviour of weapons to hit more shots (for example in CS:GO or Apex especially weapons behave differently from each other after a few bullets have been fired)
Non-Static Playstyle	3	Playing unpredictably so the enemy can’t adapt to you easily or adapting to enemy playstyles
Movement	3.4	Mechanics to make yourself harder to hit or play more unpredictable using movement quirks in the game engine (bunny-hopping in CS:GO, sliding/running/jumping in Apex)
Performing Under Pressure	3.49	Performing as well as you would in usually while being aware of circumstances such as being the last on your team alive

Figure 3: Outcome of question in mechanics survey asking players to rank mechanics from a filtered list of those found in Figure 1.

3.2.3 Requirements

Comparing the two research approaches, behaviours like crosshair placement and positioning seem to be the most popular mechanics players think need to be improved. Aim in general is also viewed as quite integral to improvement in ranking. The project will likely focus on these 3 mechanics due to the likelihood of being able to implement the complexity required and the generic nature of the mechanics making it easier to evaluate the tool relative to existing alternatives. Other close in popularity mechanics, like game-specific utility, are too complex for it to be feasible to implement due to variance between games. A table was then made outlining each identified requirement in Figure 4.

Figure 4: Requirements Identified and Priority

Requirement	Notes	Priority
1. Basic FPS with simple movement, shooting and targets	Player should be able to move around an environment and shoot targets that react in some way.	Must
2. Change engine to feature mechanics that are planned to be trained	• Make shooting/damage similar to other popular tactical shooters for low time to kill • Inaccuracy while moving • Consistent recoil pattern • Damage given to enemies reliant on where bullet lands	Must
3. Create basic game around these mechanics	• Design map • End game when player has a specific number of kills • Give a score based on performance • Make basic UI to display info to player • Create basic menu to start and close game	Must
4. Track player performance to influence DDA	• Overall accuracy • Headshot percentage • Time to damage/kill • Health loss per enemy	Must
5. Adjust enemy difficulty based on previous performance	• Reaction time • Damage • Fire rate • Health • Movement Speed • Spawn Rate	Must
6. Create options menu	Allow player to adjust every setting for themselves or for enemies, as well as some system settings like mouse sensitivity.	Should
7. Track statistics for learning systems	• Time to damage/kill • Crosshair placement (horizontal and vertical) • Accuracy • Movement while shooting • Number of enemies exposed to • Damage taken	Must
8. Give player feedback of poor performance	After an enemy is killed tell the player what they did poorly and how to improve upon it	Must
9. Improve overall UI	Add better menus and graphics aid with the game being easy to use and help information be communicated	Should
10. Log statistics to file	Allow player to output their performance to a file to be analysed at the end of the study	Could
11. Custom Gamemode	Allow the player to customise the gamemodes within the trainer to help it seem closer to whatever commercial FPS title they wanted to train for, assuming a system like this was not implemented within the game itself	Won't

3.3 Initial Testing of FPS Prototype

As an initial test of how feasible the FPS prototype plan was, the Unity FPS Sample project was downloaded and set up, and the work environment configured [26]. After confirming the project worked as expected, the code was edited to confirm it was possible to make the changes that were expected to the engine. Headshots (damage multiplier based on shot position) and a UI indicator for such shots were added as part of this. This was evidence of being able to influence/listen to events in the game engine, and provide feedback to the player upon these events. Example screenshots can be seen in figures 5 and 6.

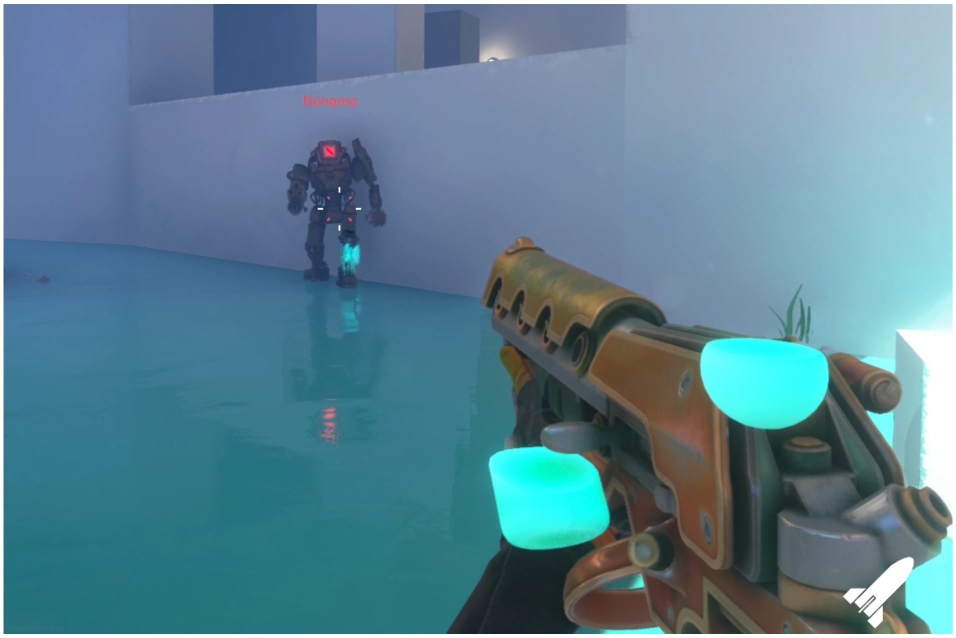


Figure 5: Bodyshot and normal damage indicator

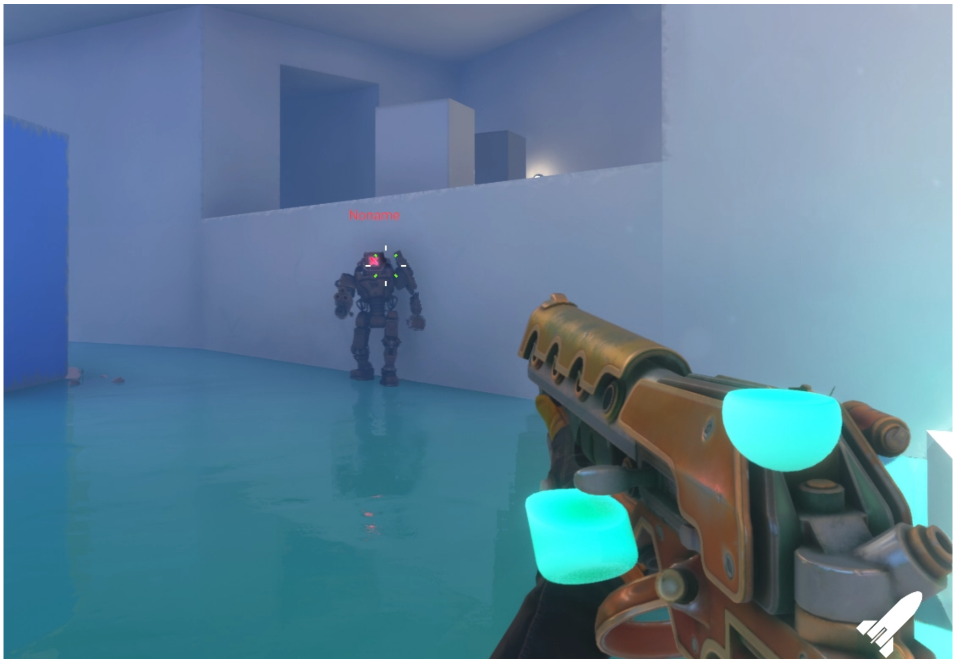


Figure 6: Headshot causing extra damage and using different indicator

This was later changed once proper requirements were outlined and attempted to be implemented, due to the complexity of the FPS Sample's inbuilt networking systems.

3.4 Plan for Evaluation

The game should be evaluated by 10+ FPS players of varying skill levels. A survey will be designed asking qualitative questions about their experience with the learning feedback and the dynamic difficulty. This will try and gauge the effectiveness of the flow implementation on learning and how they felt the mid-game feedback impacted their improvement, as well as asking them specifically how they felt about the methods of feedback.

A comparative evaluation will not be used due to this prototype focusing on experimenting with feedback content and methods mid-game. Due to the amount of time it takes to definitively improve over time at a game compared to how much time is available for the trial it is unlikely such a comparison would yield interesting results on its own. If implemented, their performance can also be submitted through the survey to be quantitatively analysed alongside the qualitative feedback.

4 Implementation

4.1 Creation Process and Prototype Overview

4.1.1 Base Game (Requirements 1-3)

Unity was selected to use for developing the game and the learning system due to prior experience, the wealth of support provided for it online through various forums and the potential for support within the University as the tool is taught in the Game Design module at Southampton. It was also likely to have basic tutorials/existing projects that could be built on top of. After further experimentation with the initial FPS sample, it proved too complex and undocumented to be suitable for the project after all, as it had a large focus on networking which was unnecessary for a single-player project. Using the old sample would have required removing many features in the demo, and the time taken to understand it would have been significant. It was replaced with an FPS from an online tutorial that featured basic movement, such as running, jumping and sliding, several weapons, and targets that reacted upon being shot [27] [28]. This was then modified to replace the existing mechanics and align them closer to that of CS:GO or Valorant [18] [22]. This includes:

- Adjusting damage to take only 2 headshots/3-4 body shots to kill a target
- Greater inaccuracy while moving
- Systematic recoil pattern of the weapon
- Removing all guns but the basic one, that is most like the common weapons from those games
- Disabling unneeded custom movement and animations

Enemies were created using dummy-shaped models adjusted to hold a weapon with hitboxes for each appendage, allowing for positional damage to be implemented [29]. They were then programmed to fire at the player upon spotting them with a random amount of time between each bullet within a range, and a health system was added to the player (shown in Figure 7).

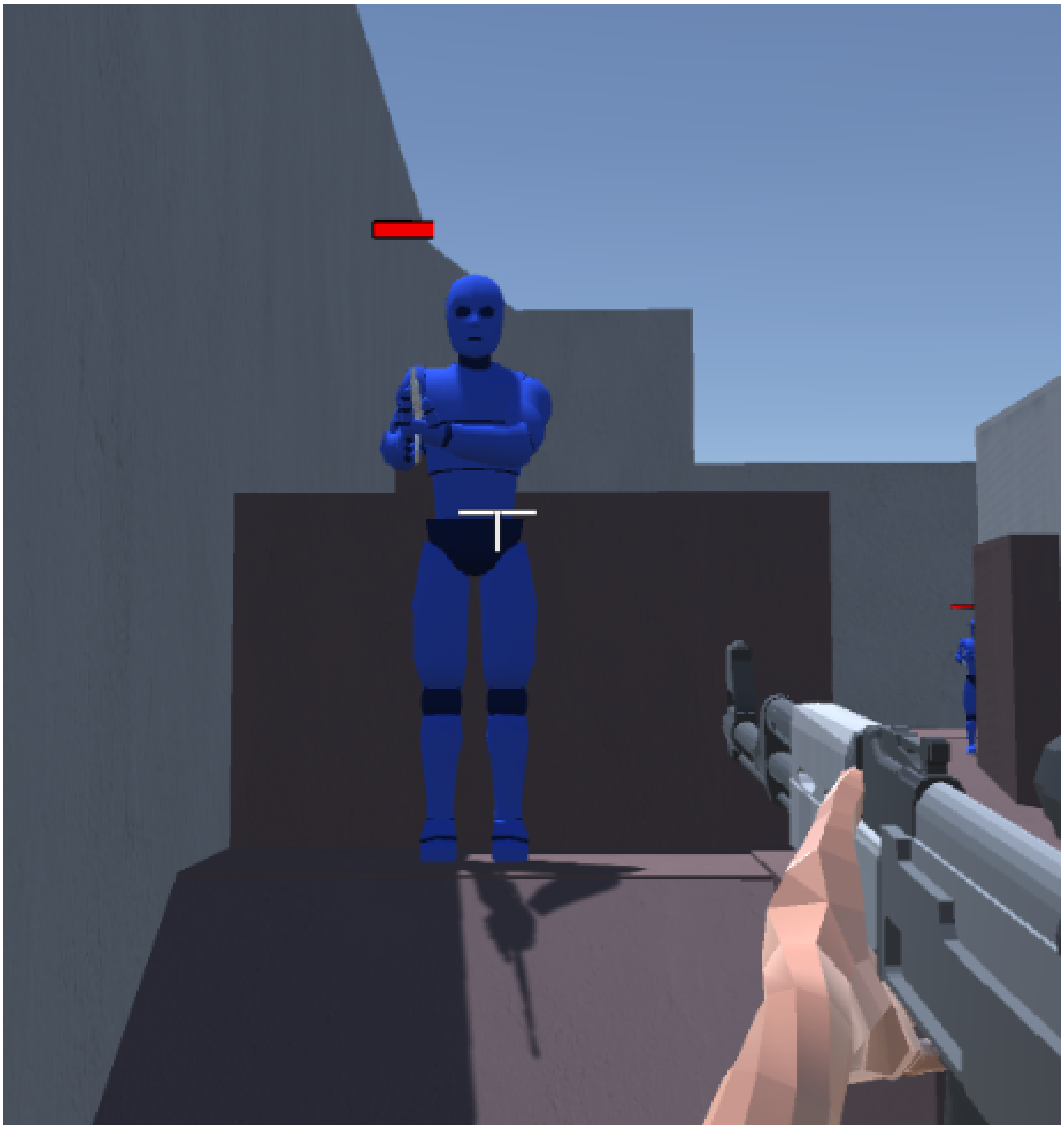


Figure 7: Example of enemy in game.

The basic game idea is similar to other aim trainers, like the YPRAC practice modes, where the player must travel through an environment defeating enemies who are hiding and waiting for them to pass; this gameplay style is very similar to what a player would be doing in a PvP tactical shooter [19]. A map was designed (see Figure 8, and Figures 18 and 19 in Appendix B for wireframes) created to facilitate this, focusing on creating a linear environment for a player to pass through with isolated angles that could be approached one at a time in order to ”clear” them and take out enemies, without exposing themselves to more enemies than necessary. Spawn points for enemies were manually created around this map, and an enemy would randomly appear somewhere currently not in view of the player at the start of the game and when one was defeated it would also spawn another enemy.

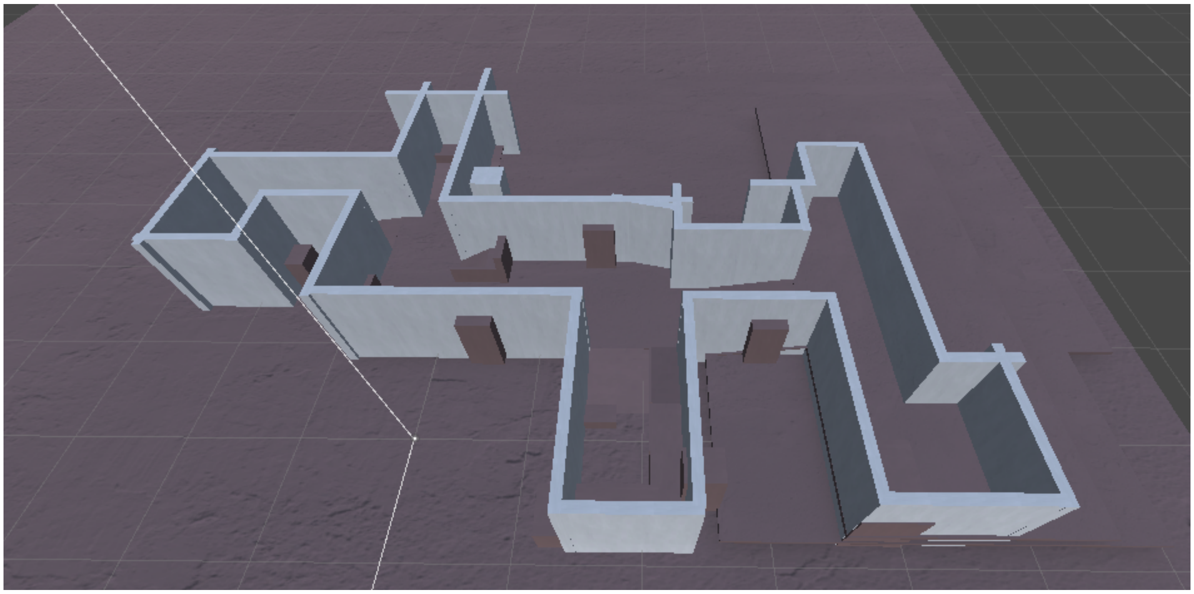


Figure 8: Map in Unity Editor.

This was then tied together with a main menu, results screen giving them a score and a time taken, and UI for the player showing their health and kill count mid-game (wireframe sketch in Figure 21). The player would spawn in and have to kill 20 enemies as efficiently as possible (activity diagram in Figure 9).

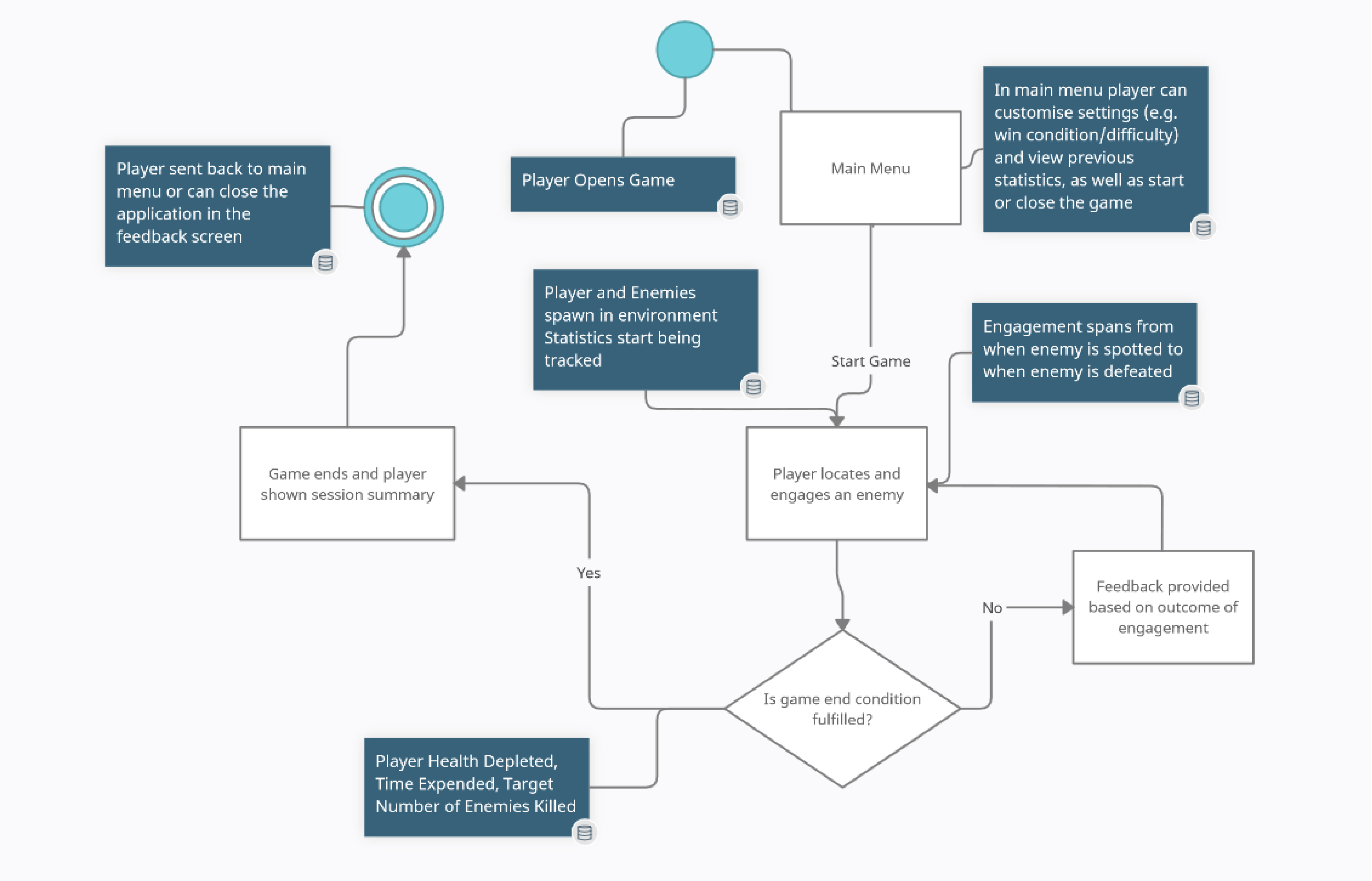

Player Opens Game
Main Menu
In main menu player can customise settings (e.g. win condition/difficulty) and view previous statistics, as well as start or close the game
Player sent back to main menu or can close the application in the feedback screen
Player and Enemies spawn in environment Statistics start being tracked
Start Game
Engagement spans from when enemy is spotted to when enemy is defeated
Game ends and player shown session summary
Player locates and engages an enemy
Yes
No
Feedback provided based on outcome of engagement
Is game end condition fulfilled?
Player Health Depleted, Time Expended, Target Number of Enemies Killed

Figure 9: Activity Diagram showing planned usage of game.

4.1.2 DDA (Requirements 4 and 5)

To try and keep the player engaged and induce a state of flow, enemy attributes like damage or health are adjusted per-round according to the player's performance. A player's performance is judged based on 6 main factors:

- Accuracy
- Headshot Percentage
- Time to Damage (Upon being spotted by the enemy)
- Time to Kill
- Kills per Second
- Health Lost

A "level shift" system was implemented to adjust the difficulty based on these factors. The player plays 5 "calibration" rounds and the 6 factors are recorded, and after these 5 rounds they should have adjusted any difficulty settings they wanted manually so they are at a level they feel comfortable at. They then are assigned a starting level of 20 (chosen as it is low but high enough for them to possibly go down from their starting point too). After any future rounds, for each factor being tracked, their previous 5 rounds worth of statistics are normally distributed, and the result from the current round is compared to that sample. The result is given a score based on how it compares to those previous results, and multiplied by 10. If the player "won" (defeated 20 enemies) then an additional 40 points are added, creating a score between 0 and 100. If they scored 80-100, the "level shift" is 2, 60-80 then 1, 40-60 then 0, 20-40 then -1 and 0-20 then -2. At the start of the next round an attribute specific constant multiplied by the level shift is added to the enemy attributes, meaning a positive level shift would increase damage and health making the enemies more difficult, and vice versa (see planning document/visual explanation in Figure 20).

The attributes adjusted for the enemies include:

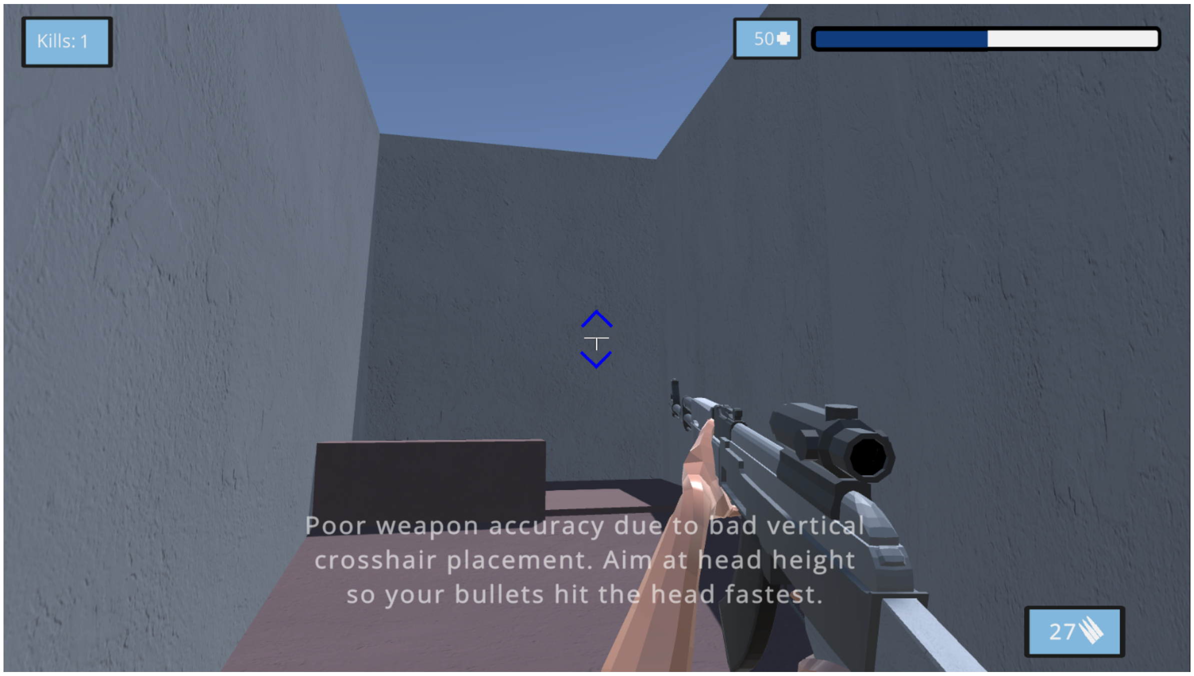
- Reaction Time
- Maximum/Minimum Fire Rate
- Accuracy
- Health
- Damage
- Number of Enemies Present at One Time
- Movement Speed

4.1.3 Feedback Systems (Requirements 7 and 8)

After an enemy spots the player, an "engagement" starts and statistics are tracked. When the player kills the enemy, the statistics are analysed and the system shows the player the mechanic they executed the poorest and why. The player has done something

”poorly” if their statistic is lower than the ”target” statistic. The thresholds for these targets were based on values found for mid-rank players on Leetify [20]. As Leetify has a large dataset of stats like crosshair placement or time to damage, they seemed like a reasonable source of initial targets. What the player has done the poorest is then chosen based roughly on the complexity of the task. For example, not moving while shooting or not entering an engagement with low ammo has a fairly simple solution and so is prioritised. As they both involve quite binary decision making before shooting (e.g. do I have enough ammo? yes/no) it is likely to be resolved quickly, whereas with more complex skills like crosshair placement it will take more practice to solve and so is expected to be shown more often. Only one subpar mechanic is chosen per engagement due to the design of the feedback UI (see Figures 10 and 23).

The feedback given to the player is a mixture of graphical and textual feedback. This textual feedback is based on the WoWAnalyzer style of feedback, where the player is told what they did wrong, how to fix it and why it is important [16] (Mindmap of how/why relationships in Figure 11). The graphical feedback is in the form of a small icon appearing near the crosshair, with the shape of which indicating what they did poorly specifically (e.g. example crosshair placement) and the colour indicating what it affected (e.g. headshot percentage). Both of these feedback methods are shown to a player for 2 seconds following an engagement.



Poor weapon accuracy due to bad vertical crosshair placement. Aim at head height so your bullets hit the head fastest.

Figure 10: Example of the UI following an engagement.

4.1.4 Optional Requirements (Requirements 6, 9, 10 and 11)

Due to extra time after implementing the functionality, a menu where the player could customise almost every aspect of the virtual mentor was added. This would allow them to calibrate the game to a level they found comfortable and enjoyable by changing the enemies' attributes. The player could also disable or adjust the thresholds for most parts of the feedback system, disable textual or graphical feedback, and disable the dynamic levelling system. This philosophy is from the game fl0w mentioned in the literature, where the player choosing their own level allows them to fine tune their experience to maximise enjoyment [2].

The UI of the game was also improved. A colour scheme was chosen, royalty free music was added in the background, and the menus were given colours and backgrounds and minor animations to improve the feel of the game [30] [31]. Examples of planned UI improvements can be found in Figure 22. The functionality to export a log of all the player's game sessions was also added to allow some qualitative data to be evaluated.

Requirement 11, while it would be helpful for the trial so players could imagine it in the context of the game they would want it to be implemented in, is somewhat out of the scope of the project due to its' complexity. As the prototype is evaluating the system itself with the idea that it could be implemented into future games as an extension to their tutorials, giving players the ability to customise the trial is not a priority.

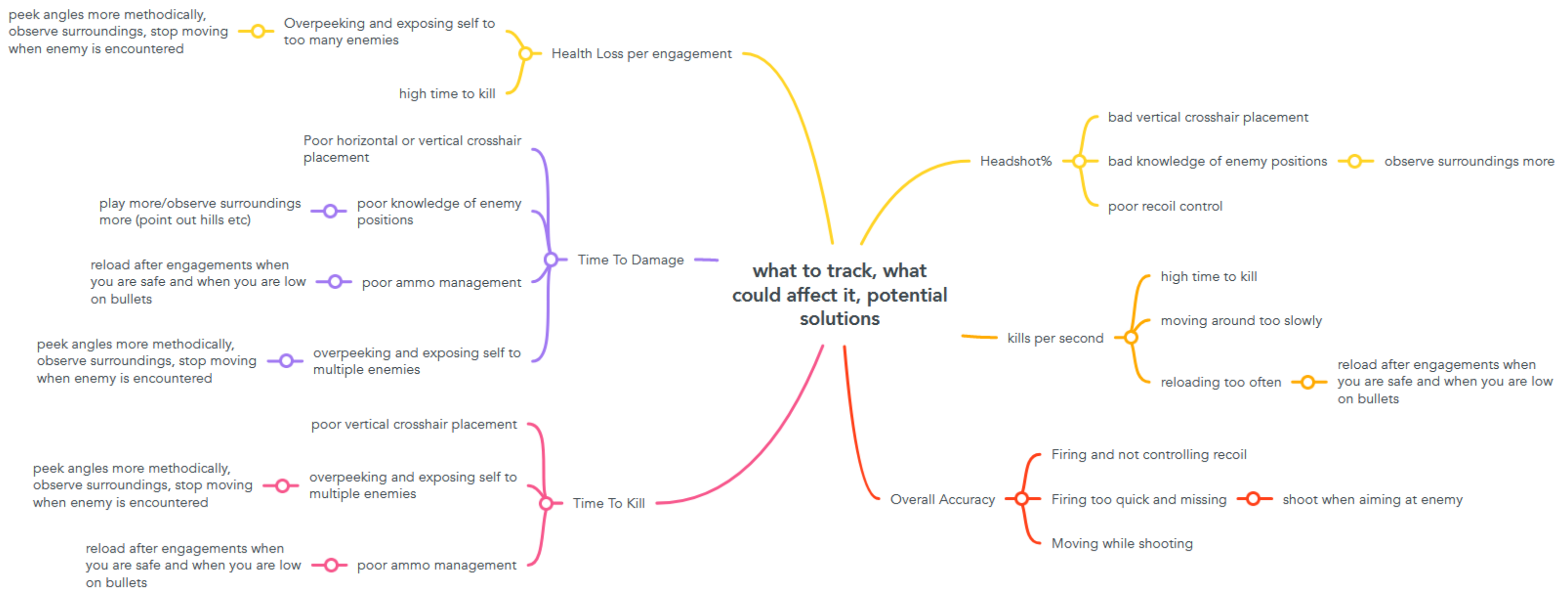

what to track, what could affect it, potential solutions
Health Loss per engagement
Overpeeking and exposing self to too many enemies
peek angles more methodically, observe surroundings, stop moving when enemy is encountered
high time to kill
Time To Damage
Poor horizontal or vertical crosshair placement
poor knowledge of enemy positions
play more/observe surroundings more (point out hills etc)
poor ammo management
reload after engagements when you are safe and when you are low on bullets
overpeeking and exposing self to multiple enemies
peek angles more methodically, observe surroundings, stop moving when enemy is encountered
Time To Kill
poor vertical crosshair placement
overpeeking and exposing self to multiple enemies
peek angles more methodically, observe surroundings, stop moving when enemy is encountered
poor ammo management
reload after engagements when you are safe and when you are low on bullets
Headshot%
bad vertical crosshair placement
bad knowledge of enemy positions
observe surroundings more
poor recoil control
kills per second
high time to kill
moving around too slowly
reloading too often
reload after engagements when you are safe and when you are low on bullets
Overall Accuracy
Firing and not controlling recoil
Firing too quick and missing
shoot when aiming at enemy
Moving while shooting

Figure 11: Mind map of planned how/why relationships between mechanics.

5 Project Evaluation

5.1 Testing

Due to time constraints, a lack of prior experience with the libraries and the existence of other suitable tests, Unity's in-built automated testing libraries were not utilised; however, whenever a major change was pushed a series of actions in the prototype to try and verify that from start to finish it worked as intended was used. This adapted as the prototype developed, but finally involved:

- Clearing all player preferences.
- Playing through a round of the game, attempting to manually trigger each type of feedback by purposefully making the mistake.
- Noted damage values of enemies and time to kill to make sure all of the default values were still reading correctly (e.g. 10 damage at first playtime).
- Noted results screen, making sure all values were within expected bounds.
- Going to the options menu and disabling/changing several attributes of the enemy and the player targets.
- Playing another round and noting if settings changes had impacted the game as expected.

More in-depth testing was also developed at the start of working on each requirement, to try and quantify when it was implemented enough to be considered "finished". This testing was deemed sufficient in place of unit testing as it increased the likelihood of finding any bugs that would be found during an average experience, and ensuring all features were working as intended, while also allowing the "feel" of the game to be tested and allowing the experience to be tuned as it was developed, seen in Figure 12.

Req. No.	Test
1	Player should press play, walk around environment without issues, shoot targets, and verify targets respond appropriately.
2	Player accuracy should be lower while moving, recoil pattern on weapon firing should be almost the same every time, shooting enemies in the head should deal more damage than in the body, and more in the body than in the arms/legs.
3	Player can open the game, press play on the main menu, defeat 20 enemies, and be shown a results screen with accurate information about kills obtained, and repeat, then close the game from the main menu.
4	Console should print accuracy, headshot percentage, time to damage and health loss at the end of each round. Player should play two rounds and maximise then minimise these statistics by missing intentionally etc. and see if results are accurate.
5	Player should use Unity PlayerPrefs to manually set enemy attributes, and play a round with attributes ”harder” and one with attributes ”lower” to verify they have an effect on the gameplay. Then reset the PlayerPrefs to the default and enable the DDA, and play several rounds while printing the attributes to make sure they are increasing as they play well/decreasing if they don't.
6	Player should use options menu to change every attribute potentially customisable (such as enemy health, or disabling graphical feedback in the learning system). Manually verify for each option that it has the desired effect in game by printing values for each, going to the menu and changing them, then loading another round and printing the values.
7	Console should print accuracy, headshot percentage, time to damage/kill, enemies exposed to, health loss and crosshair placement after each ”engagement”. Player should maximise and minimise these statistics manually by playing and verify values change as expected.
8	After each engagement the player should be shown text related to the statistic they were performing the worst in. Play rounds performing poorly in each tracked attribute (crosshair placement, etc.) to verify that the feedback that appears feels fair.
9	Ensure menu has not lost functionality by testing previously tested but changed sections of the menu.
10	Player should export .json file from the options menu and manually verify that it contains the information required (all details from each engagement, from each round played and the settings at the start of the round).
11	Not Implemented

Figure 12: Requirements and their respective ”completion” goals

5.2 Evaluating Prototype

A survey was created to obtain feedback for the prototype [32]. It asked quantitative questions about the player's broad feelings on the experience to get an idea for how appropriate people felt the prototype was and to see how much it succeeded as a ”game”, but was mostly focused on qualitative questions for praise or issues on both the Dynamic Difficulty Adjustment system and the Feedback System. The questions asked the players to describe their experience with the dynamic difficulty adjustments, and any particular issues they had with it, and then asked about their experience with the quality and format of feedback given, and how useful different methods were to them. Responses to the quantitative questions should help verify if the players felt the game succeeded at being a good base for the learning system, and the qualitative question responses will help identify which elements of the learning system are good and which need more refinement in the future.

The survey was then sent out to University of Southampton students who were known to play FPS games, asking them to play the prototype for roughly 10 minutes a day for a week (also following the guidance for the module-level ethics approval, ERGO/FEPS/41626.A2).

At the end of that week they completed the survey and gave their thoughts on the experience, as well as a log of their game data for potential quantitative analysis. 14 participants took part in the trial, however only 13 ended up responding to the survey, of which 2 for some reason only finished the first page. Due to the anonymisation of the survey data it was impossible to locate those who had signed up yet not completed the survey, and they did not complete it after several public reminders.

Once the results were collected, the qualitative feedback was generalised and grouped in such a way that it could be tallied. These groups of feedback were then coded and sorted by theme (Figure 13).

All participants felt they improved over the course of the trial (and did - as can be seen in Figure 14, their relative difficulty levels improved over time) with a majority mentioning they felt the feedback was useful and the DDA helped them stay engaged. Most participants mentioned the pictures being the easiest feedback to take in. Only one player disagreed that the game was mechanically comparable to existing FPS games but they clarified later that they meant it was ”too close to Counter-Strike and Valorant”, which is what the mechanics were modelled after.

Some other feedback into how the UI could present information, like the horizontal/vertical crosshair indicators giving the specific direction as well as just horizontal or vertical (e.g. left or right, up or down), or having a passive head height indicator, was suggested to support players in learning things like where to line up their crosshair, in contrast to just telling them to position their crosshair better.

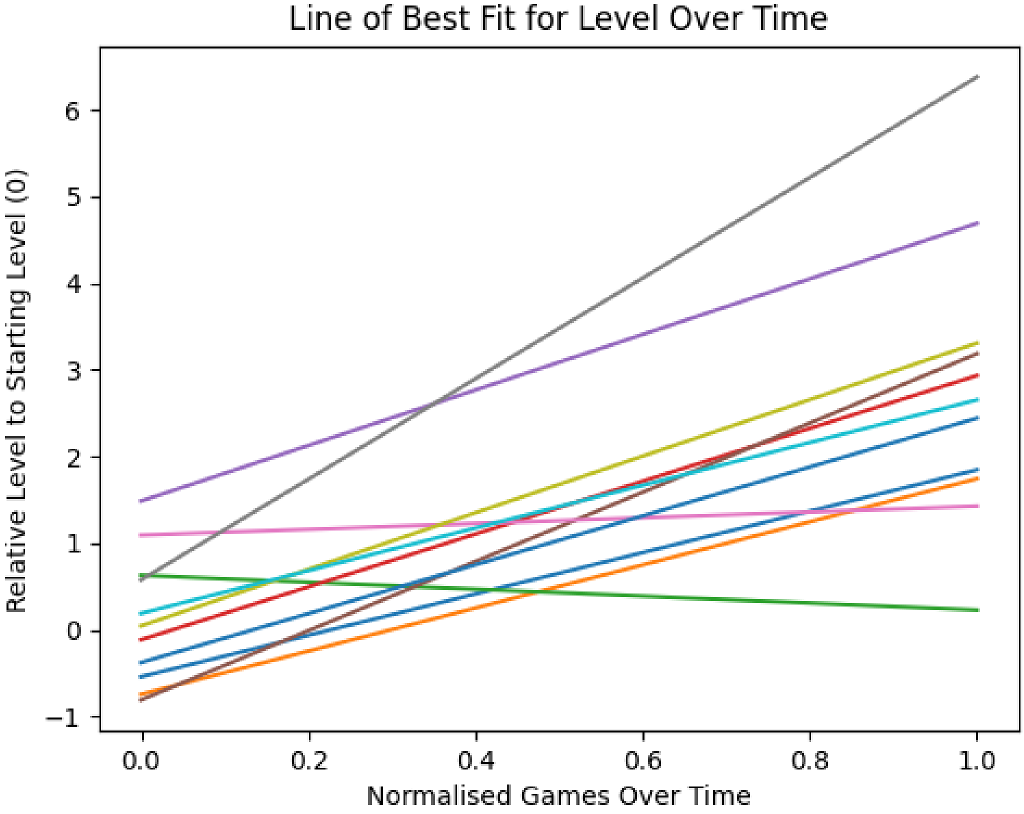
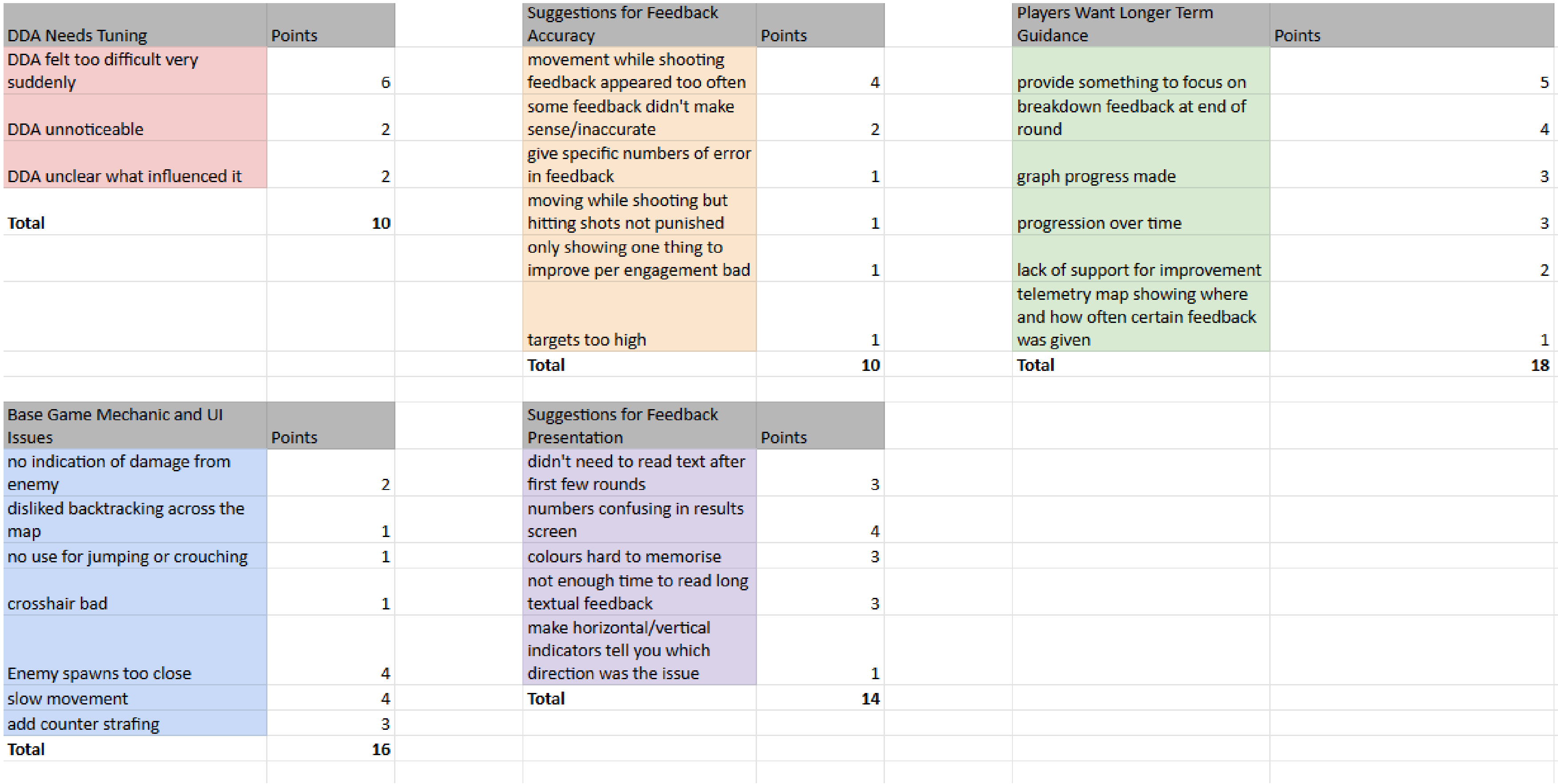


Figure 14: Best Fit for games played over time per user and the players' relative skill level.



DDA Needs Tuning	Points
DDA felt too difficult very suddenly	6
DDA unnoticeable	2
DDA unclear what influenced it	2
Total	**10**

Suggestions for Feedback Accuracy	Points
movement while shooting feedback appeared too often	4
some feedback didn't make sense/inaccurate	2
give specific numbers of error in feedback	1
moving while shooting but hitting shots not punished	1
only showing one thing to improve per engagement bad	1
targets too high	1
Total	**10**

Players Want Longer Term Guidance	Points
provide something to focus on	5
breakdown feedback at end of round	4
graph progress made	3
progression over time	3
lack of support for improvement	2
telemetry map showing where and how often certain feedback was given	1
Total	**18**

Base Game Mechanic and UI Issues	Points
no indication of damage from enemy	2
disliked backtracking across the map	1
no use for jumping or crouching	1
crosshair bad	1
Enemy spawns too close	4
slow movement	4
add counter strafing	3
Total	**16**

Suggestions for Feedback Presentation	Points
didn't need to read text after first few rounds	3
numbers confusing in results screen	4
colours hard to memorise	3
not enough time to read long textual feedback	3
make horizontal/vertical indicators tell you which direction was the issue	1
Total	**14**

Figure 13: Themes identified from feedback (points = unique mentions).

5.2.1 Long-Term Guidance

The most common theme in the feedback was the idea of long-term guidance. Players noted not being able to reflect on the feedback they obtained mid round, meaning they had to memorise the feedback in the couple of seconds it was shown and remember that they had to work on it in future rounds. While there were miscellaneous averages of their statistics shown in the results screen, there was not any information on the specific feedback received, so they would not be able to see the how or why the statistics were below targets.

For example, players thought it would be good to be told ”what [they] had done the worst in that run”, allowing them to identify a weakness more clearly than just the numbers provided by the results screen. A ”post-round breakdown” of the round was also requested a few times, showing something like the details of each engagement and the judgements made on it. Features like this were not focused on in the prototype due to them being less in-game than the implemented learning system, making it a lot more like online tools like Leetify where you just see a summary, though clearly these features are more of a baseline than expected. A potential implementation to go with this would be a replay of the engagement which would allow the player to re-experience the engagement in isolation and learn from mistakes made they might not have noticed, or see why the learning system gave the feedback it did. This would allow the player to dive more into the mistakes they made, while not losing the ”in-game” feel of the improvement advice that this prototype aimed to deliver to distinguish itself from existing solutions.

Players also wanted more of an idea of the feedback they were receiving over time, not just over that round, so it would be easier to compare their performance to their ”bests or the bests of the last 5”, or get a ”graph of progress made”. This would likely add to the feeling of improvement the players would be able to achieve as they could properly visualise the statistics given, as well as help them see what they on average need to focus on more clearly. One player proposed a ”telemetry map” of data, mapping types of feedback to positions on the map which alongside the replay feature idea could potentially help a player associate things they need to improve on with a specific location in the map and helping them recall it.

5.2.2 Feedback Accuracy

While in general players felt that the feedback provided mid-game was useful, some exceptions arose that they felt detracted from the experience. The most common complaint was about the ”moving while shooting” feedback, which multiple players felt appeared too often/unfairly. This was left out of the options menu, as unlike the other feedback factors there was not a target that was thought to be adjustable as the mechanic fired based on the velocity of the player, however this by mistake also meant players couldn’t disable the feedback. This is potentially because there would be residual velocity when the players moved that was small enough to not be noticeable or have a noticeable effect on the bullet spread (which was also tied to velocity), yet was still above 0 and so prioritised over any other mistakes. Fixes for this include making the movement feedback fully customisable, including a velocity threshold for what constitutes movement that the player can select, or potentially allowing the player to prioritise their feedback without disabling any entirely. As the current feedback system attempts to not overload the player with information it only provides what it prioritises as their most important mistake for any given engagement.

The feedback in this group was a lot more specific and mentioned slightly less than other sections. Some specific feedback mentioned ”moving while shooting but successfully hitting the shots not being punished” which, is a flaw in the learning system counting ”moving while shooting” as a ”why” the engagement was bad, so without an associated ”how” it was bad it would not suggest anything was wrong even if technically the player was doing something ill advised. There were also complaints about a lack of specific numbers in the feedback (as in, ”you were facing 50 degrees too high” rather than just ”poor vertical crosshair placement”), which could potentially be useful to help contextualise the extent of the mistake to the player so they know how much to adjust. Finally a couple of players also disliked the ability to only see one item of feedback per engagement. In a future prototype the UI design could be improved to allow this to be implemented cleanly, and would be worth comparing in the future, but the post-round breakdown mentioned in the previous section could provide this as well.

Some also felt that the targets were too high, and while they could be customised, the targets could potentially be set based on the players' averages alongside the difficulty. The pre-selected targets, while based on research, were too general to apply to all players so in the future scaling the targets to be a flat increase to their average would give them a dynamic target to work towards rather than an end goal.

5.2.3 Feedback Presentation

Players noted several potential flaws in the way the feedback was presented to the player. The most common observation was that the post-round results screen was ”too confusing” as the numbers were not contextualised, there was purely a list of their averages for things like crosshair placement compared to the their set targets. Players were likely confused as while they could compare the targets to their averages, those numbers did not have any meaningful link to what they were doing in the game, so it did not actually help them know what to do to improve. In future post-round feedback screens giving visual examples of the issues, such as a freeze frame of their crosshair placement with a diagram showing the offset, would allow the player to learn the meaning of the numbers rather than assuming they would understand them straight away.

Another issue with the feedback presentation was that players found that they ”didn't need to read the text every time". As the text was always accompanied by a specific icon/colour, and there was little variance in the text due to not giving specific statistics just overall advice, players were associating the block of text with the icons that appeared and from then on the text was only taking up space on the screen. It was noted that it was "helpful at the start to help remember what the icons mean", so the textual feedback was not actively detrimental, and players were also able to disable it if they ever felt it was too obtrusive. This suggests having the text get gradually less detailed/obtrusive after the player uses the tool more would improve their experience. However, several players also noted they "never fully memorised the colours" despite a colour code being accessible at all times by pressing tab, suggesting that the reinforcement of text worked a lot better with the icons which conveyed more information on their own than the colours did, and one player found that the colours had to be "reacquainted" whenever you played again. This implies that graphical feedback was more useful for rapid feedback than textual, but the icons themselves were the useful part rather than the colour code due to the icons having shapes with a recognisable meaning that didn't need to be learnt. More evidence supporting this is in players saying that the "red and green was a good indicator at a glance" for the results screen, as red and green have pre-existing connotations with failure and success that the player did not need to learn. In the future both the why and the how should potentially be presented with their own graphical icons, with the colours being an aid alongside feedback rather than being the only feedback.

Players also found that due to the fast-paced nature of the engagements sometimes they would "not have enough time to read the feedback in time for each". This also suggests that the graphical feedback is more useful to players over time as the meaning can be understood much more quickly than reading a paragraph. Either spacing out the enemies so there is more time between each engagement or adding a queue system for the feedback so that prompts from previous engagements aren't lost while you continue to play the game would benefit the player's experience.

5.2.4 DDA Tuning

The dynamic difficulty adjustment present in the game was designed to help the player stay engaged and get into the "state of flow" that would improve their learning abilities. As the level was tied to their own relative skill after the 5 calibration rounds, for every player they should fluctuate around a difficulty level where they win about half of the time which is similar to what matchmaking for competitive games aims to achieve (giving both teams as close to 50% chance to win as possible).

Most players agreed that the game was of an engaging difficulty with or without any customisations, however the most common observation from the trial as a whole was that it often felt like there was a "difficulty spike" after a few levels where it became a lot harder compared to the gradual increase of previous levels. This could potentially be due to the scaling of the abilities being linear with caps, as opposed to something more quadratic or logarithmic that would allow attributes like reaction time to increase by less as time goes on, as a jump between 0.8 and 0.9 seconds reaction time feels different to a jump from 0.7 to 0.8. Another issue with the linear scaling is that the constants selected may have been inappropriate relative to the others, for example damage of enemies increasing and enemy health increasing is a lot more of a noticeable difficulty jump as it will start taking less time for you to lose and more time to kill enemies, whereas with attributes like enemy accuracy or reaction time the adjustments are much more subtle. Finally, enemy speed was not relative to the distance they had to travel which could vary, meaning that some enemies at high speeds would randomly move a lot faster than other enemies nearby due to having a further distance to travel in the same time. This could be fixed by having a base speed be constant rather than have the enemy travel a specific distance in a specific time and have it be multiplied by the difficulty adjusted speed.

One idea proposed by a player that could be worth considering is tying the difficulty increases to specific attributes rather than based on your averages as a whole, for example if your tracking average was good then the enemy speed would increase but the health would stay the same, increasing health if players get better accuracy, and increasing fire rate as the player's time to kill improves. This could prevent a feeling of a difficulty spike as not every attribute would be increasing in difficulty at the same time, and also be supporting the learning techniques of the game as every aspect that is measured they would be challenged at according to their level in that aspect.

The other complaints about the dynamic difficulty were mostly that it was unclear what influenced it, as there was no real in game explanation, just a given score and a level that either increased or decreased. Having a tooltip explaining the level system or redesigning it to be more intuitive (e.g. starting at 0, giving a breakdown of what factors went into the level change) would allow the players to feel like they knew what was influencing their level.

5.2.5 Underlying Game Issues

The base game was designed for the purposes of this prototype, and as it was not the priority for the experiment not as much design work was put into the details as would be expected in a commercial competitive shooter. Some bugs, like enemies not entirely following the laws of gravity or the shoot button sticking occasionally, were unable to be fixed by the time of the trial and so participants were asked to focus their feedback on what seemed like a design choice (or a lack thereof), rather than bugs and lack of variety in the gameplay.

Players disliked ”enemies spawning behind them”, especially in line of sight and thus shooting the player. The fact that enemies were able to spawn and be able to shoot you immediately is unintended, and possibly suggests a disjoint between the ”spawning an enemy” and ”shooting a player” line of sight logic from a given position. While technically the fact that enemies can spawn behind you does emulate modern tactical shooters, where players need to be conscientious of places they have been in the past, that is also more relevant on less linear maps. As the map for this prototype was linear there is technically no way an enemy could get from somewhere not yet cleared to somewhere cleared without the player having spotted them. For the purposes of training the map could either be extended so the player did not need to "backtrack" to areas they already had cleared, or the map could be non-linearly designed so that the backtracking is worked into the gameplay and the player can loop around rather than following the same line back and forth.

Some other complaints by players were about the movement feeling too slow, and not being similar enough to "counter-strafing" from CS:GO where the player can cancel their velocity entirely by tapping the movement key for the opposite direction, allowing them to immediately be fully accurate. Most of these issues are just due to the base project used for movement, and it is hard to gauge exactly how fast the players would want to move. One idea that would allow players to "tailor the trainer towards different FPS titles" would be to give the player proper control over the mechanics as well as just the attributes, so things like player speed and gravity would be customisable and they could make it feel like the game they wanted to play. As mentioned in the earlier requirements, customisable gamemodes to tailor it to specific FPS titles was considered, however was not prioritised as theoretically this system could be an extension to a tutorial in a real competitive FPS rather than a standalone game like Aimlabs.

5.3 Project Management

5.3.1 Tools Used

Tools were used to aid development of the prototype throughout the project. The game engine used was Unity, due to prior experience using it, its user-friendly interface and the vast amount of users it has allowing open-source projects like the one used as a base for the movement in the prototype to exist. It also can connect with the JetBrains IDE Rider, an editor specifically for C# that integrates with Unity with no setup. As JetBrains provide student licenses for their IDEs, the availability of both was useful. To track progress, provide version control and keep backups of the code safe, the project was uploaded to a private GitHub repository.

For planning and reporting on the project, alongside handwritten planning documents, Creately and Paint.net were used to create graphics and charts. Overleaf was used for writing the actual report as it provided version control and hosted the report online, also protecting against local data loss. For the evaluation survey Qualtrics was used due to the University providing a license, and it allowed for anonymous data collection (so the trial fell under generic ethical permissions for the project), file uploads, and a useful interface for viewing and exporting the survey data. Excel was also used for analysis of the data, and creating the tables seen in Figures 1 and 13, and Python was also used for analysis of player data from submitted .json files, creating graphs such as the one in Figure 14.

5.3.2 Risk Analysis

Likelihood and Severity of risks was rated on a scale of 1-5 in Figure 15.

Risk	Likelihood	Severity	Total Risk	Contingency Plan
Data Loss	2	5	10	Work will be saved in the cloud via tools like Github or Overleaf to prevent local issues resulting in loss
Insufficient Technical Skill to Complete Planned Work	2	4	8	Work plan involves choosing a mechanic, implementing it into the FPS project, then implementing tracking for that mechanic, and finally implementing a visual aid to help work on that mechanic. That way if I fail to implement a mechanic I can move onto the next mechanic, or if I fail to implement an aid I will still have implemented the mechanic and the tracking for it. Hopefully through this the risk will be minimised as there will still be work to analyse.
Insufficient Number of Participants for Evaluation	2	3	6	I will obtain ethical approval to allow players outside the University to be surveyed to test the tool, which increase the number of potential applicants. I can also evaluate the tool by critically comparing it to other methods and existing tools which can help supplement a poor sample size.
Personal Reasons or Other Work Impacting Time I Can Dedicate To Project	4	2	8	I can always adjust the Gantt chart, with the earlier method of developing I mentioned the chance of there being something to evaluate is maximised, and if there is a suitable reason for the time management change I can also apply for deadline extensions.
Loss of Equipment	1	5	5	Due to the graphical intensity of the game it will be hard to develop the project if I lose access to my desktop PC. I would need to make use of the University Labs, provided they are open.

Figure 15: Risk analysis for the project

5.3.3 Gantt Chart

The project timeline did not entirely follow the planned progress of the first Gantt Chart (Figure 16) due to January being dedicated to exams and personal issues in February preventing work on the project. With all that ongoing, realising that the base FPS originally selected for the project before Christmas would not be suitable also impacted on motivation. However, an extension was obtained and once the decision to start the practical implementation afresh was made the project moved swiftly, represented in Figure 17. Moving the initial Gantt chart back by a month to account for the extension, the timings somewhat line up with the initial plans, and the prototype was finished with a month to spare for evaluation and report write-up.

If the project was to be re-done then the biggest change made would be more vigorously testing the Base FPS before finally selecting it, or evaluating more options before attempting to implement features. The decision to change the Base FPS was delayed due to the fear of losing progress, which ended up wasting more time overall. Working on motivation and having the ability to assess the situation and re-prioritise early on would be beneficial for ensuring the project stayed moving. More time should also have been dedicated to the project during the exam season, as ultimately the revision was not as demanding as expected.

With more time for the project, more time could also be invested into working out small issues/fine-tuning the experience with the learning system, and securing ethical approval for evaluation outside the University, which in turn would allow for more likely participants and a control group to actually try and measure quantitatively the performance of players and how that changed over the trial period. A longer trial would also allow for players to get more experience with the tool, and have better ideas about what they liked/disliked about the system.

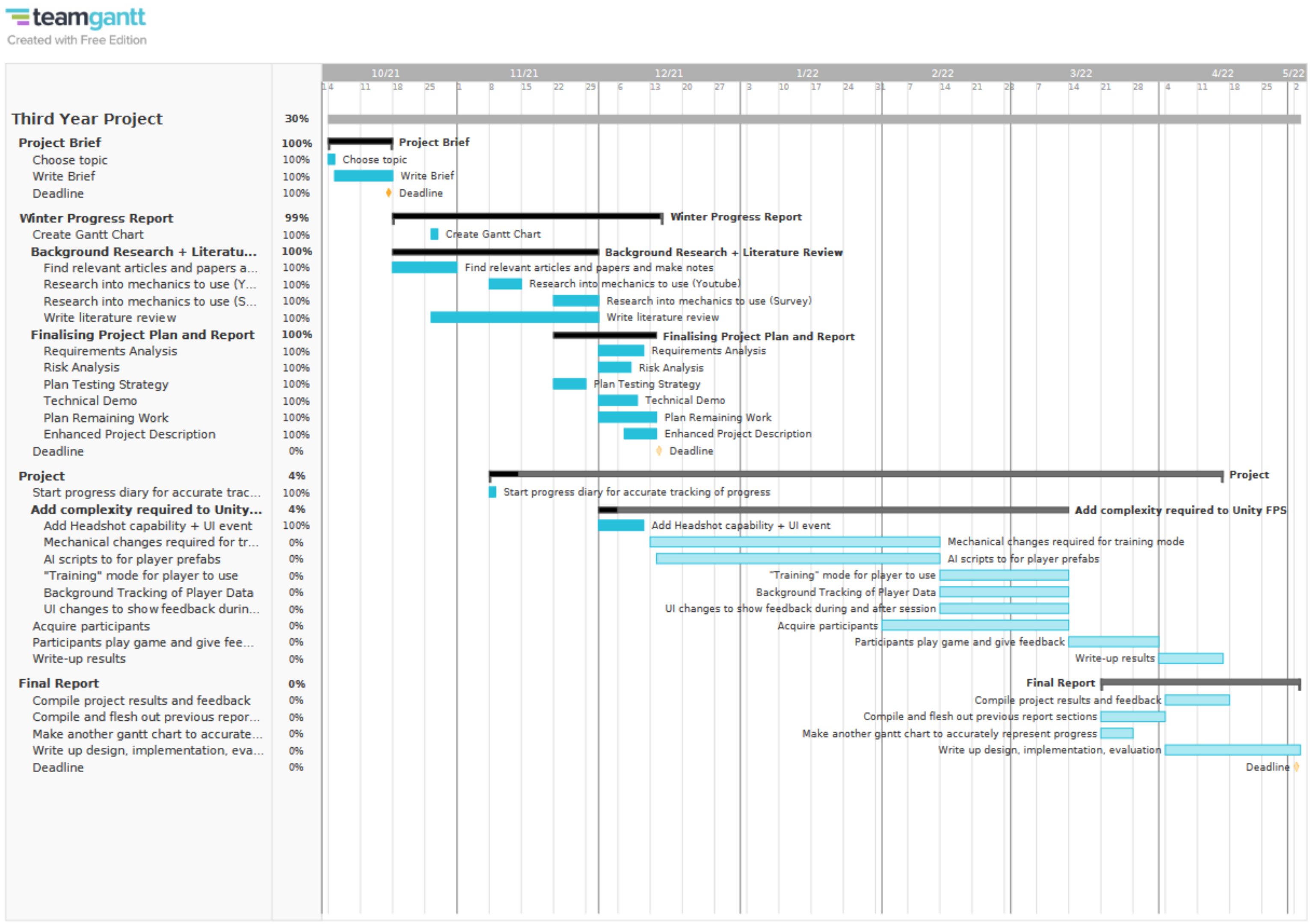


Figure 16: Gantt Chart at the Interim Report Deadline.

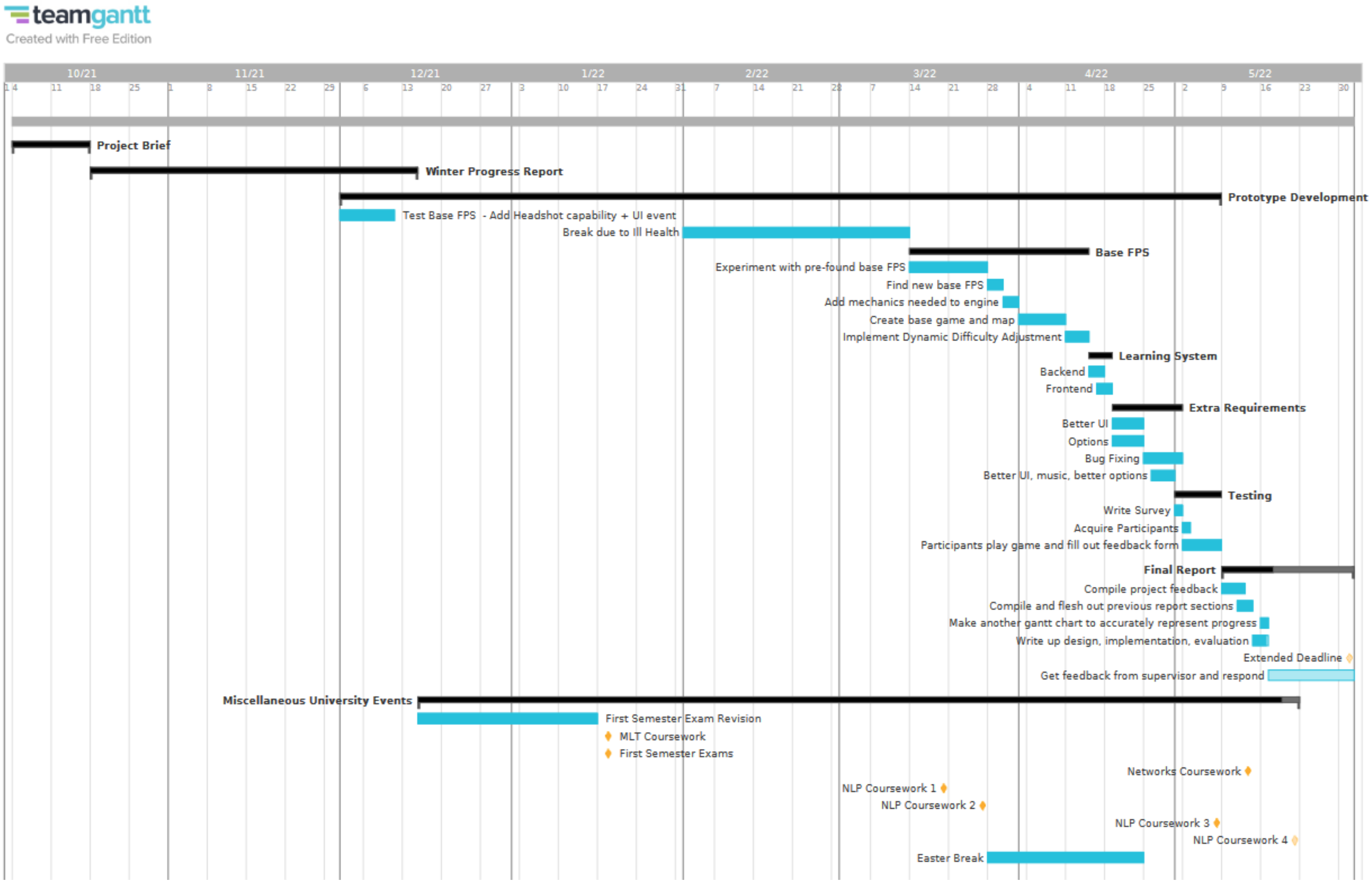


Figure 17: Gantt Chart at the End of the Project.

6 Conclusions

6.1 Summary

This project aimed to explore the effectiveness of in-game mentoring in tactical FPS games past the point of standard tutorials. A prototype FPS was made with dynamic difficulty adjustment allowing the player to stay engaged and induce a state of flow over many repeated rounds of the training game. A learning system providing live feedback to the player while they were in this state of flow was then implemented, tracking the player's performance across a variety of mechanics from popular tactical FPS games, telling them what they did wrong, how to improve and why it was important.

The prototype was then tested by University of Southampton students, who played the prototype across a week long period and completed a survey at the end. The survey was then thematically coded, identifying the most common groups of feedback presented by the players. It was generally found that players enjoyed the combination of live feedback and dynamic difficulty allowing themselves to track their improvement, however the main issue they found was with tracking their progression over time as feedback was only presented per-engagement. Players also did not like the prioritisation of the feedback, as that was unable to be customised. Icons were preferred to colours and text for feedback as they were more easily understandable in the midst of fast-paced gameplay.

The players thought this method of in-game learning was effective and stood out compared to existing ”aim trainer” style statistics pages or mini-games, and so could potentially be implemented as an extension to popular FPS games successfully. Future work would be needed to refine the way feedback is given and prioritised, to balance between providing the player with all the information they want but not overloading the screen or giving too much information to be taken in. The dynamic difficulty adjustment also needs to scale more fairly so players don't feel like the difficulty ”spikes”.

6.2 Limitations and Future Work

The tool developed during this project only featured one level, on top of a basic game made from scratch. There was no real variance in gameplay, and while players did feel it was similar it was still an abstracted version of commercial tactical FPS games, and as such will feel different to the games participants would be used to. A future implementation could feature more avenues for customisation, giving them control over the physics and game engine to some degree to allow for variations to aspects like player speed/accuracy, or changing the environments and objectives of the levels, to allow the learning game to feel more like the specific game that they would want to be training for.

A comparative evaluation would also be useful for providing evidence of feeling improvement compared to playing without the feedback system. A larger-scale trial could be undertaken, tracking several people of similar ranks in one game (for example, silver in CS:GO), with one group training with the learning system tool and one group training just by playing the game itself, and over a period of time (perhaps a month?) tracking the increase in their rank relative to their playtime in the game. This on top of a qualitative insight into how they felt would allow the effects of the tool to be properly evaluated.

The tool was also limited as the feedback that it provided was still somewhat general, only providing the player with the area that it thought they performed the worst in. With more time invested in UI design a lot more information could be passed on to the player each engagement, allowing more learning to occur as they play without obscuring the screen or being too difficult to understand, as well as the results screen being more understandable. The feedback could also be more dynamic, giving specific numbers or directions to guide players' actions so they know what to do to improve, or even providing guidance before the end of an engagement, through UI indicators showing the optimal height to aim for example.

For some games, like CS:GO, there is an amount of customisability possible through the development tools released, so it could be possible to implement a system similar to this directly into a custom level in the game. This would reduce the amount of time needed to tweak the base of the tool to feel like a specific game as the work would already be done, however it would likely be difficult to implement as publicly released developer tools will have a lot less support available online.

6.3 Concluding Remarks

In this project I set out to explore the potential for more involved feedback for tactical FPS games to allow players to learn and improve more efficiently without a human coach, as most only teach you the basic mechanics and systems of the game and from there players are required to learn through observing other players in multiplayer games, or go to external community tools to analyse their gameplay and tell them vaguely what they are doing wrong. While it did not have a comparative evaluation, I believe the tool created in this project was a valuable experiment in taking the dynamic, specific feedback utilised in tools like WoWAnalyzer, and transferring it to competitive tactical FPS games [16]. Players of low and high skill levels found the tool enjoyable and thought the feedback methods were interesting and effective, and the players did all improve over their time with the game relative to the difficulty setting it calibrated for them. I believe the next best place for this area to go would be to either try and implement such a system within an existing game engine, such as for CS:GO, or to flesh out the variety of the base gameplay and fine-tune the feedback so that players get all the information that they want to keep track of their improvement over time and know what to work towards.

7 Bibliography

References

[1] Jeanne Nakamura and Mihaly Csikszentmihalyi. Chapter 18 - Flow Theory and Research, The Oxford Handbook of Positive Psychology. Oxford University Press, 2009.

[2] Jenova Chen and Nicholas Clark. Flow - thatgamecompany, 2007. `https://thatgamecompany.com/flow/` (accessed: 25.10.2021).

[3] Ian Schreiber. Decision making and flow theory, 2009. `https://gamedesignconcepts.wordpress.com/2009/07/20/level-7-decision-making-and-flow-theory/` (accessed: 25.10.2021).

[4] Arttu Perttula, Kristian Kiili, Antero Lindstedt, and Pauliina Tuomi. Flow experience in game based learning – a systematic literature review. International Journal of Serious Games, 4(1), 2017.

[5] Alena Denisova, Paul Cairns, Christian Guckelsberger, and David Zendle. Measuring perceived challenge in digital games: Development & validation of the challenge originating from recent gameplay interaction scale (corgis). International Journal of Human-Computer Studies, 2019.

[6] Supat Sanjamsai and Darunee Phukao. Flow experience in computer game playing among thai university students. Kasetsart Journal of Social Sciences, 39, 2018.

[7] De Liu, Xun Li, and Radhika Santhanam. Digital games and beyond: What happens when players compete? MIS Quarterly, 37(1):111–124, 2013.

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

[9] Mohammad Zohaib. Dynamic difficulty adjustment (dda) in computer games: A review. Advances in Human-Computer Interaction, 2018, 2018.

[10] Nintendo. Mario kart 8, 2014. `https://www.nintendo.co.uk/Games/Nintendo-Switch-games/Mario-Kart-8-Deluxe-1173281.html` (accessed: 15.11.2021).

[11] Valve. Left 4 dead, 2008. `https://web.archive.org/web/20090327034239/http://www.l4d.com/info.html` (accessed: 15.11.2021).

[12] Simon Demediuk, Marco Tamassia, William Raffe, Fabio Zambetta, Florian Mueller, and Xiaodong Li. Measuring player skill using dynamic difficulty adjustment. ACSW '18: Proceedings of the Australasian Computer Science Week Multiconference, 2018(41), 2018.

[13] Simeon Atanasov. Exploring and designing around experience of feedback in video games, 2013. Master's Thesis at Malmö University.

[14] Rachel Jug, Xiaoyin Jiang, and Sarah Bean. Giving and receiving effective feedback. Archives of Pathology and Laboratory Medicine, 143(2), 2018.

[15] Jamie Madigan. How video games do feedback well (and poorly), 2019. https://www.psychologyofgames.com/2019/01/how-video-games-do-feedback-well-and-poorly/ (accessed: 16.11.2021).

[16] Martin Hols. Wowanalyzer, 2017. https://wowanalyzer.com (accessed: 16.11.2021).

[17] Martin Hols. Wowanalyzer contribution guidelines, 2017. https://github.com/WoWAnalyzer/WoWAnalyzer/wiki/Suggestions (accessed: 16.11.2021).

[18] Valve. Counter-strike: Global offensive, 2021. https://blog.counter-strike.net/ (accessed: 17.11.2021).

[19] Yesber. Yprac practice and warmup, 2016. https://steamcommunity.com/workshop/filedetails/?id=740795413 (accessed: 17.11.2021).

[20] Leetify. Leetify, 2021. https://leetify.com/ (accessed: 17.11.2021).

[21] Ubisoft. Rainbow 6 siege, 2021. https://www.ubisoft.com/en-gb/game/rainbow-six/siege (accessed: 17.11.2021).

[22] Riot Games. Valorant, 2021. https://playvalorant.com/en-gb/ (accessed: 17.11.2021).

[23] Aim Lab. Aim lab, 2021. https://aimlab.gg/ (accessed: 17.11.2021).

[24] KovaaK. Kovaak's aim trainer, 2021. https://store.steampowered.com/app/824270/KovaaKs/ (accessed: 17.11.2021).

[25] Youtube Team. An update to dislikes on youtube, 2021. https://blog.youtube/news-and-events/update-to-youtube/, (accessed: 20.05.2022).

[26] Unity Technologies. Fps sample, 2018. https://unity.com/fps-sample, (accessed: 12.05.2022).

[27] Colanderp. Unity tutorials, 2020. https://github.com/Colanderp/UnityTutorials (accessed: 12.05.2022).

[28] Jose Díaz. Inverse kinematics, 2017. https://assetstore.unity.com/packages/tools/animation/inverse-kinematics-1829, (accessed: 12.05.2022).

[29] Kevin Iglesias. 3d character dummy, 2020. https://assetstore.unity.com/packages/3d/characters/humanoids/humans/3d-character-dummy-178395, (accessed: 12.05.2022).

[30] Chosic. Royalty free background music download, 2022. https://www.chosic.com/free-music/all/, (accessed: 12.05.2022).

[31] Purple Planet Music. Royalty free music, 2022. https://www.purple-planet.com/, (accessed: 12.05.2022).

[32] Harry Nelson. Qualtrics survey, 2022. https://southampton.qualtrics.com/jfe/form/SV_8H6xLgEEJZka5PE, (accessed: 12.05.2022).

A Primary Research

A.1 YouTube Videos Watched

YouTube videos used for getting an idea of important mechanics to train:

- 10 Tips to Improve at CS:GO - voo CSGO
  `https://www.youtube.com/watch?v=eFab7sTSmEc`
- The Definitive Guide to Improving - voo CSGO
  `https://www.youtube.com/watch?v=ygyiM0Ctibo`
- INSTANTLY Improve Your MECHANICS With 10 PROVEN Tips - CS:GO - ProGuides CSGO Tips, Tricks and Guides
  `https://www.youtube.com/watch?v=KzpNwMDNVYs`
- How To INSTANTLY Improve Mechanics & Game Sense in CS:GO - Tips and Tricks - ProGuides CSGO Tips, Tricks and Guides
  `https://www.youtube.com/watch?v=q2Kcx8aVNkE`
- How to Get Good Mechanics & Improve Aim - eAthlete Labs
  `https://www.youtube.com/watch?v=5oFU1xV2uiU`
- 5 quick tips to improve your aim at any FPS - SteelSeries
  `https://www.youtube.com/watch?v=jPZXbW4wz_A`
- 5 Tips To Improve At Any FPS - Beginner Tips - How To Get Better At Shooters - BBKDRAGOON
  `https://www.youtube.com/watch?v=SSYdidhBPlQ`
- Your Best FPS Tips to Improve at Any FPS - How To Get Better At Shooters - BBKDRAGOON
  `https://www.youtube.com/watch?v=gFrieQ08wmw`
- How To Improve At Rainbow Six Siege In 8 Minutes - Shawk
  `https://www.youtube.com/watch?v=r2TvAuUJ9ak`
- HOW To ACTUALLY Get Better At Rainbow Six Siege - MatchPoint
  `https://www.youtube.com/watch?v=sJPl9IFkaNk`
- Apex Legends Tips That Actually Help You Improve - Dazs
  `https://www.youtube.com/watch?v=LtNwMWJq7hQ`
- Apex Legends Tips and Tricks To Improve! - TimProVision
  `https://www.youtube.com/watch?v=1-dXSNsJKsw`
- How to IMPROVE AIM in Apex Legends! - 5 PRO Tips/Tricks! - ImMadness
  `https://www.youtube.com/watch?v=6Ahhzh3g-GY`

A.2 FPS Mechanics Survey

Survey can be found at `https://forms.office.com/r/w4ii4LB8qr` and a summary of results can be seen at `https://forms.office.com/Pages/AnalysisPage.aspx?AnalyzerToken=wvK35yiTe6WIe84S2jxYFm49xwzo2HkU&id=-XhTSvQpPk2-iWadA62p2FHw1w2fiNdLkApxBaBZwiZUNzI`

Questions asked:

1. Please confirm you are a University of Southampton student and you have read the Participant Information Sheet and Consent Form.

2. Please confirm you agree with the following statements:
   - I have read and understood the Participant Information and have had the opportunity to ask questions about the study.
   - I agree to take part in this study.
   - I understand my participation is voluntary and I may withdraw at any time and for any reason.

3. What competitive FPS do you play most often?
   - CS:GO
   - Valorant
   - Rainbow 6 Siege
   - Apex Legends
   - Overwatch
   - Other (Insert)
   - N/A

4. What rank have you achieved in that FPS?
   - Top rank obtainable in game
   - Fairly high rank (DMG and up in CS, Diamond and up in Overwatch, etc.)
   - Middle rank (Gold in most games)
   - Lower rank (Copper/Bronze/Silver etc.)

5. In terms of mechanics or game knowledge for your most played game or FPS's in general, what would you say are the most important things you needed to improve before you were able to achieve that rank? (For example, crosshair placement, counter-strafing, knowing legend abilities, positioning, etc.) List as many as you like.

6. Of these mechanics and gameplay features found through prior research, how would you rank them in terms of importance? (Most important near the top)
   - Crosshair Placement
   - Having a non-static playstyle (reacting to enemy behaviour, not being predictable)
   - Counter-strafing or other game-specific movement techniques
   - Positioning
   - "Aim" in general (e.g. tracking or flicking)
   - Performing under pressure
   - Spray control + weapon familiarity
   - Game specific utility skills (grenades/character abilities)

B Implementation

B.1 Map Designs

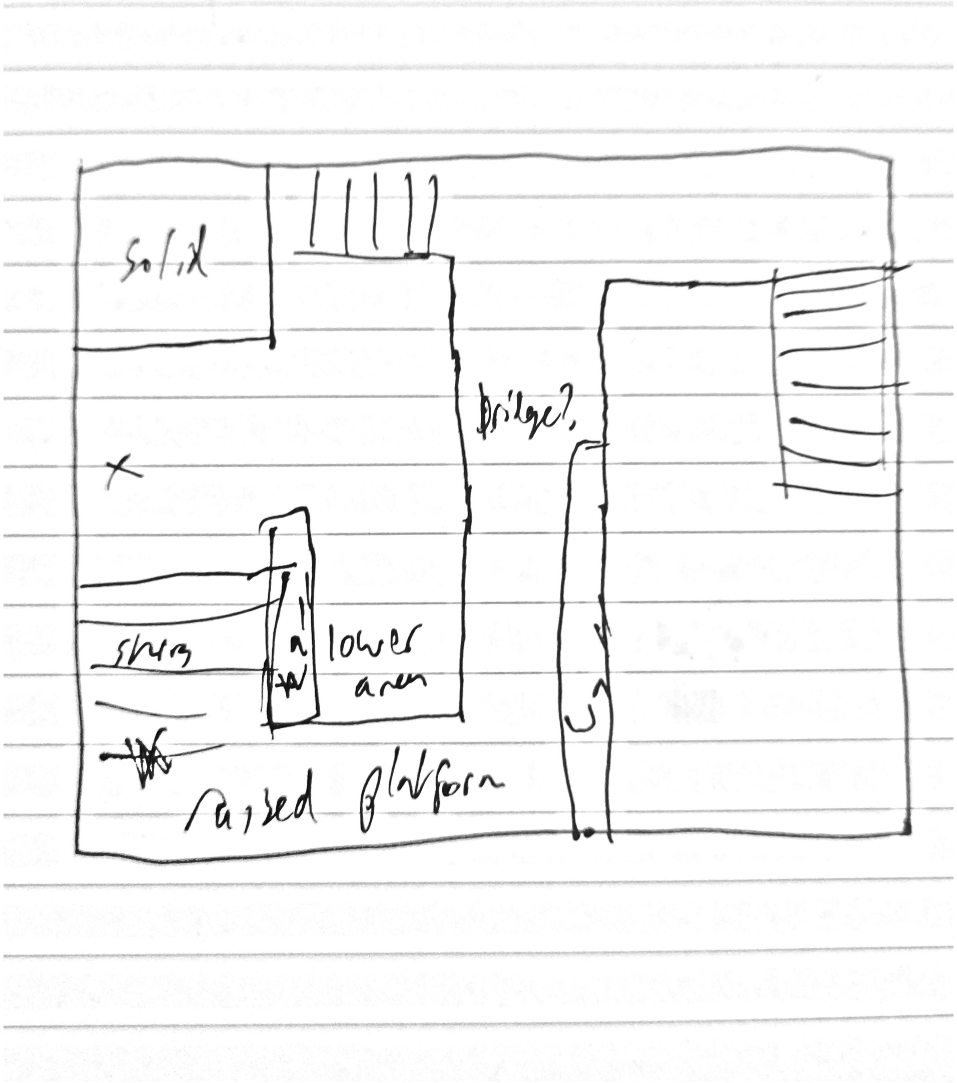


Figure 18: Initial sketch of map plan.

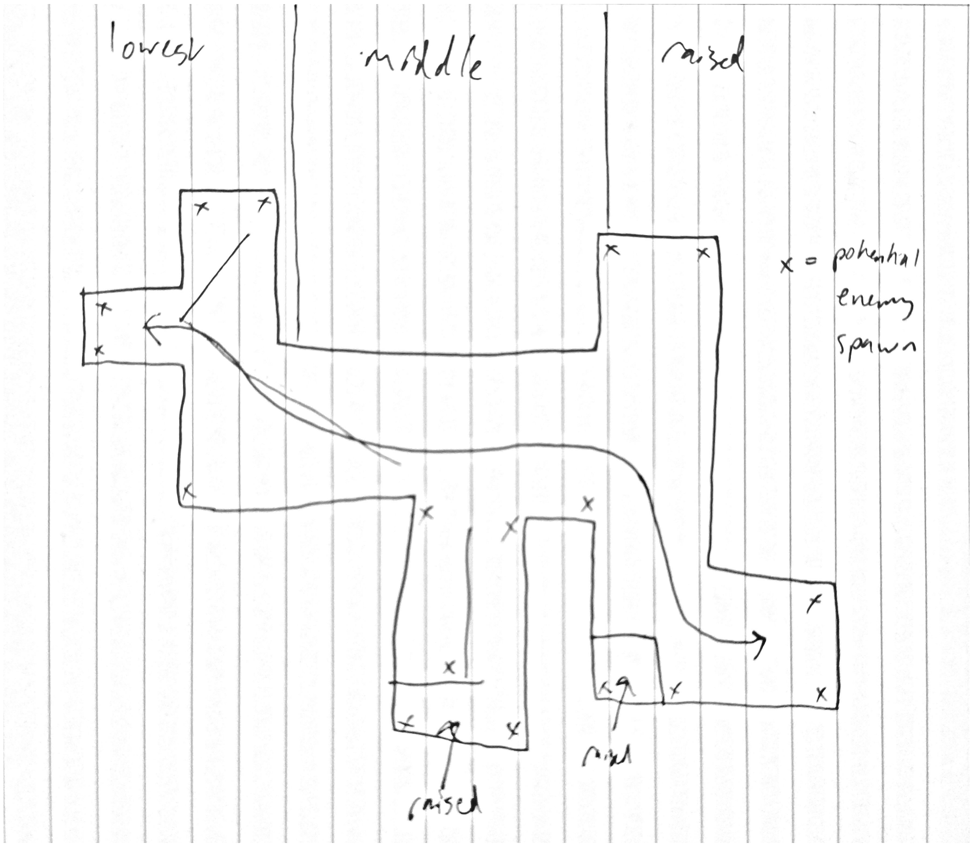
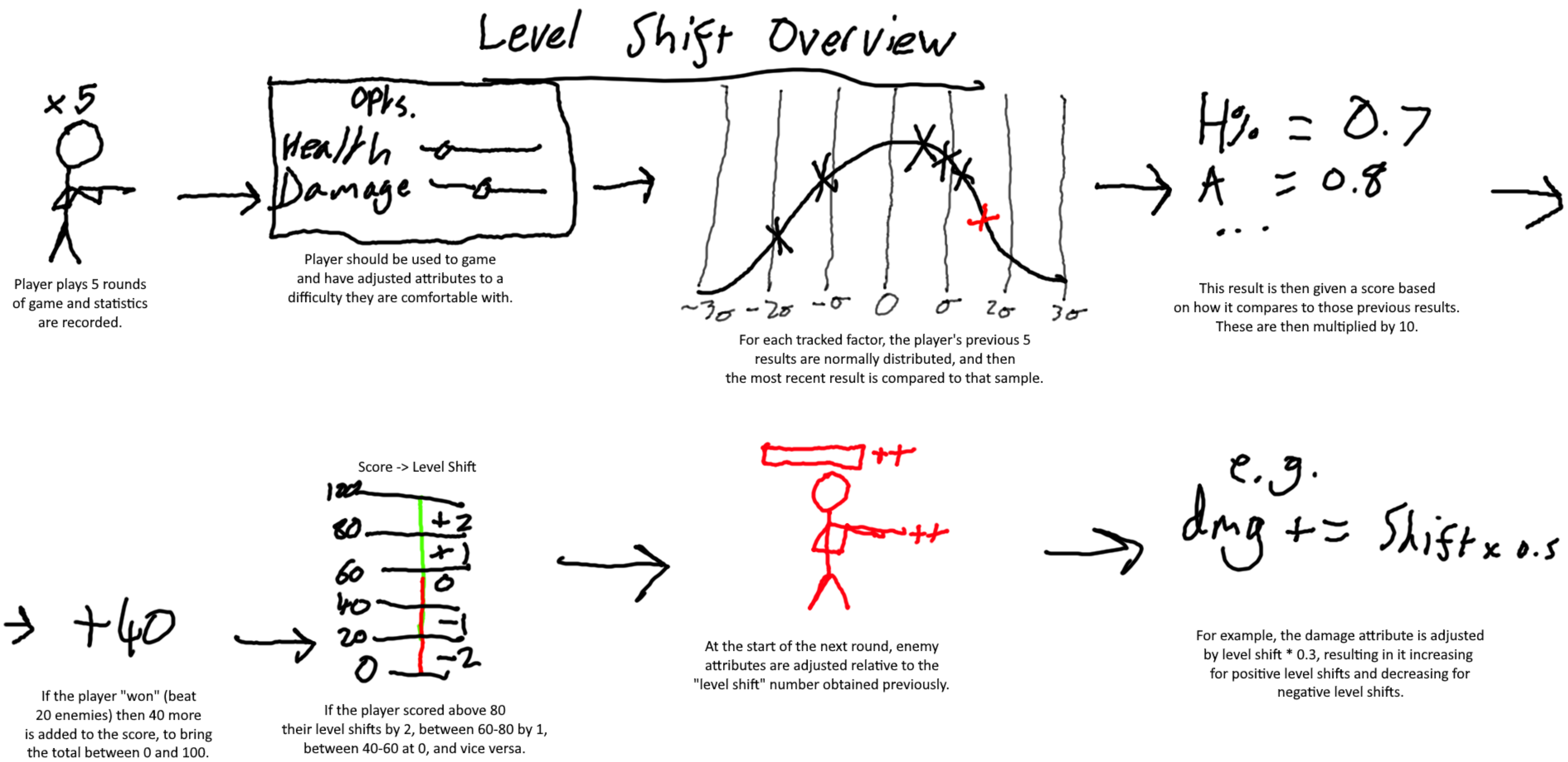


Figure 19: More linear redesign of map.

B.2 Level Shift Plan



Player plays 5 rounds of game and statistics are recorded.

Player should be used to game and have adjusted attributes to a difficulty they are comfortable with.

For each tracked factor, the player's previous 5 results are normally distributed, and then the most recent result is compared to that sample.

This result is then given a score based on how it compares to those previous results. These are then multiplied by 10.

If the player "won" (beat 20 enemies) then 40 more is added to the score, to bring the total between 0 and 100.

If the player scored above 80 their level shifts by 2, between 60-80 by 1, between 40-60 at 0, and vice versa.

At the start of the next round, enemy attributes are adjusted relative to the "level shift" number obtained previously.

For example, the damage attribute is adjusted by level shift * 0.3, resulting in it increasing for positive level shifts and decreasing for negative level shifts.

Figure 20: Planning diagram of Level Shift system.

B.3 UI Plan

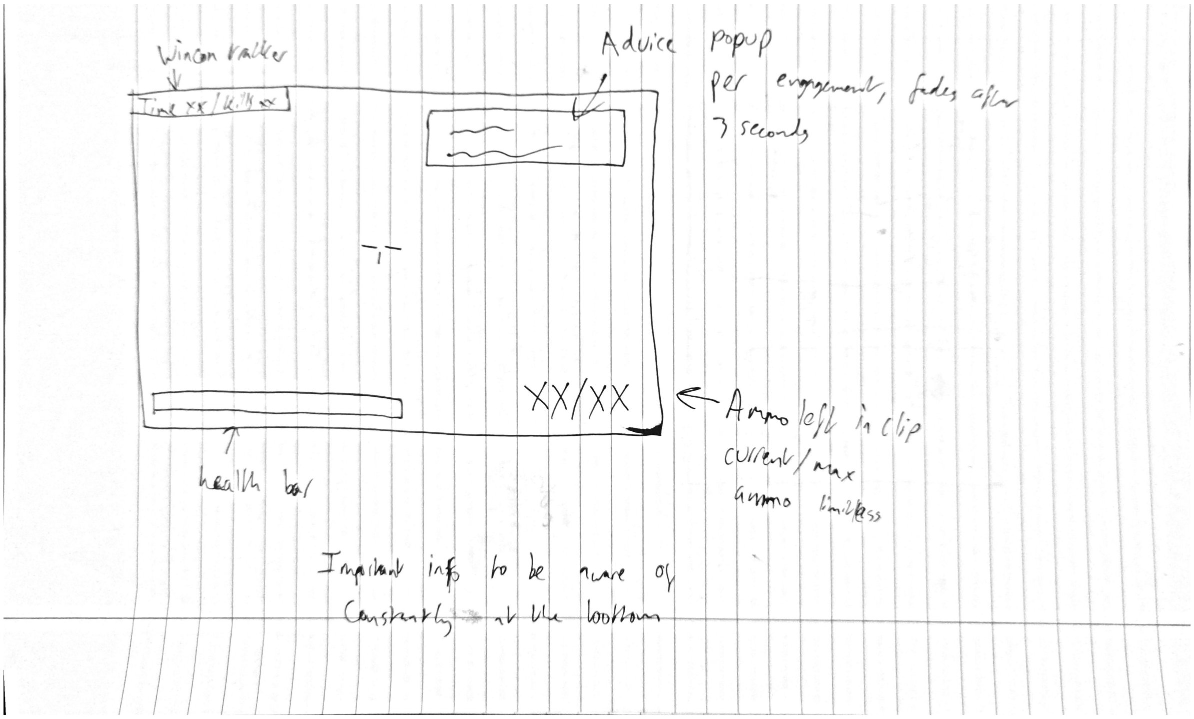


Figure 21: Wireframe of planned game UI.

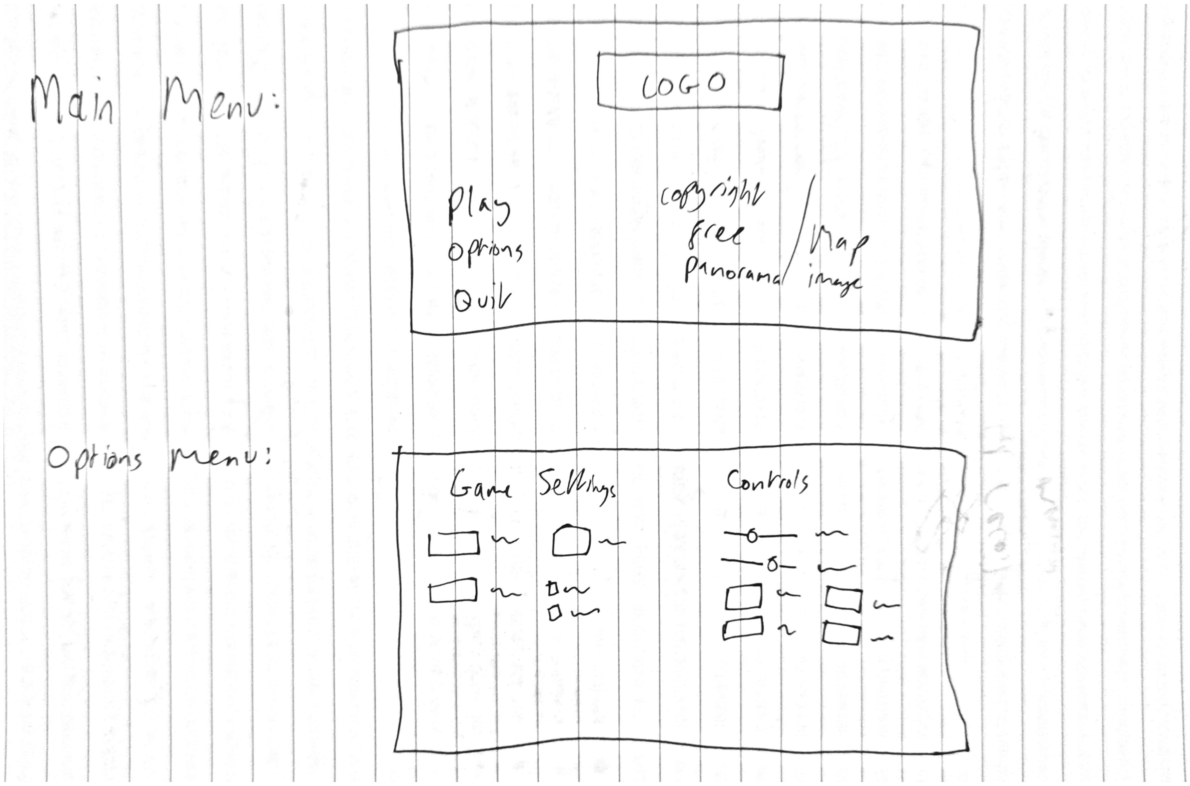


Figure 22: Planned menus.

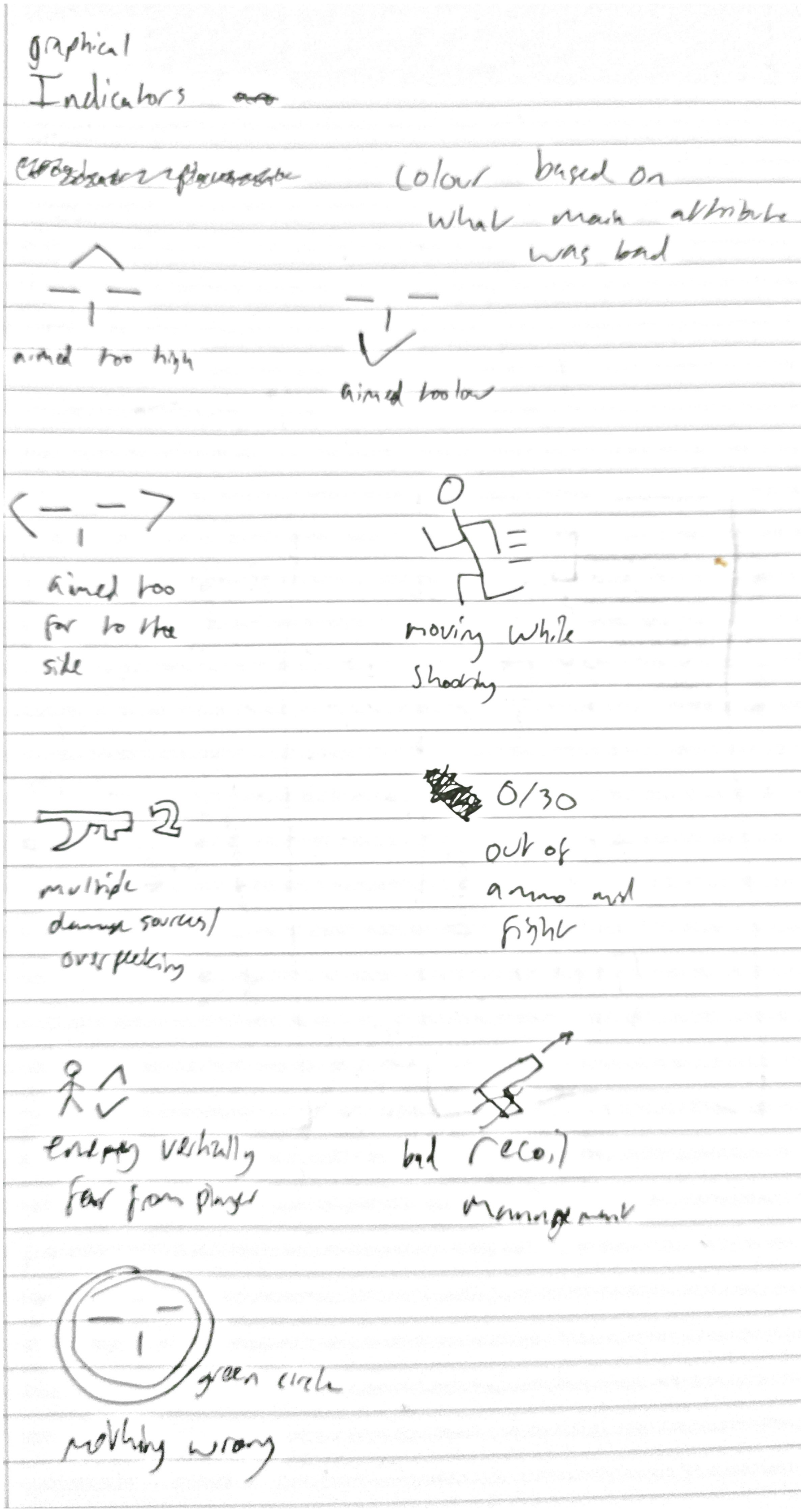


Figure 23: Planned icons for feedback indicators.

C Evaluation

C.1 Survey Questions

Questions asked in feedback survey presented to users after trial.

1. Please indicate that you have read, understand and agreed with the following statements:
    - I have read and understood the Participant Information and have had the opportunity to ask questions about the study.
    - I agree to take part in this study.
    - I understand my participation is voluntary and I may withdraw at any time and for any reason.

2. Do you confirm you are a University of Southampton student?

3. How would you describe your previous experience with FPS games? (previous ranks, hours of playtime, etc)

4. Please choose and remember 3 random words so I can identify your answers should you want to withdraw your data from the trial after submitting:

5. Do you think the game was similar enough to existing tactical shooters that experiences in this game would be mechanically comparable to currently popular competitive tactical shooter games?

6. Did you find the difficulty acceptably engaging (with or without customising it)?

7. Please describe your experience with the Dynamic Difficulty Adjustment: (Thoughts on the scaling/accuracy of when levels went up or down/did you change any settings manually etc.)

8. Did you have any particular issues with the mechanics in the game? (Preferably not bugs, I am more looking for if you think they were implemented in un-intuitive ways)

9. Do you feel you improved at the game over the trial period?

10. How did you feel about quality of the feedback provided by the learning system?

11. What are your thoughts on the graphical icons vs the textual feedback given? Is one more helpful than the other?

12. Would any other methods have been better for giving you live feedback?

13. What are your thoughts on being able to make adjustments to the thresholds/disable elements of the learning system?

14. How did you feel about the feedback given in the results screen? If it was useful for you, how?

15. Please click ”Export History” in the game’s options menu, and upload the ”GAMELOG.json” file that is created in your documents folder and upload it here.

16. Thanks for participating! If you have any other thoughts or comments on the project please leave them here.