STADIA

WPI

Catalyst - Using Hitbox Scaling for Latency Compensation in a Cloud-based Game

Adam Desveaux

Alejandra Garza

Cameron Person

James Plante

Joseph Swetz

Catalyst- Using Hitbox Scaling for Latency Compensation in a Cloud-based Game

A Major Qualifying Project Report submitted to the faculty of
WORCESTER POLYTECHNIC INSTITUTE
In partial fulfillment of the requirements for the
Degree of Bachelor of Science

Submitted by:

Adam Desveaux
Alejandra Garza
Cameron Person
James Plante
Joseph Swetz

Faculty Advisor:

Professor Mark Claypool
Worcester Polytechnic Institute

Date Submitted: May 6, 2021

Photo on title page by Moises Gonzalez on Unsplash

Abstract

Cloud gaming services have been increasing in popularity because of the convenience in rendering and computation done in the cloud. However, latency remains a challenge since all player input must be sent from client to server, rendered, then sent back before action results are seen. For our project we created a multiplayer first-person shooter game with spell-based combat and deployed it to Google Stadia. We then created a latency compensation technique by scaling enemy player hitboxes based on players' latencies, and conducted a user study to determine its efficacy. Analysis of player accuracy under different latency conditions, with and without hitbox scaling, shows an increase in accuracy with compensation for higher latency values.

Table of Contents

Abstract **1**

Table of Contents **2**

List Of Figures **4**

List of Tables **6**

1. Introduction **7**

2. Related Work **11**
2.1 Latency and Interactivity 11
2.2 Latency Compensation 13
2.3 Cloud Gaming 15
2.4 Google Stadia 17

3. Development **19**
3.1 Game Concept and Treatment 19
3.2 Primary Gameplay Mechanics and Rules 20
3.3 Level Design 21
3.4 Art 24
3.5 Sound Design 27
3.6 Technical Aspects 27
3.7 Alpha Testing 30
3.8 IQP User Study 30
3.9 Latency Compensation Technique/Implementation 33

4. Evaluation **38**
4.1 User Study 38
4.2 Setup 38
4.3 Recruitment 40
4.4 Data Collection 41

5. Results **43**
5.1 Demographics 43
5.2 Player Performance 48
5.3 Quality of Experience 62
5.4: Gameplay Survey 66
5.5 Discussion 70
5.5.1 Player Performance 70

5.5.2 Quality of Experience 71
5.5.3 Gameplay Survey 73

6. Conclusions **75**

7. Future Work **78**

8. References **80**

9. Appendices **86**
Appendix 3.A: Asset List 86
Appendix 4.A: Demographics Survey 88
Appendix 4.B: Post-Trial Survey 89
Appendix 4.C: Final Survey 90
Appendix 4.D: Catalyst Recruitment Trailer 91
Appendix 4.E: Sample Log File Excerpt 92
Appendix 5.A: Cumulative Distributions of Fireball Accuracy 93
Appendix 5.B: Quantile-Quantile Plots of Fireball Accuracy 94
Appendix 5.C: Cumulative Distributions of Lightning Strike Accuracy 95
Appendix 5.D: Quantile-Quantile Plots of Lightning Strike Accuracy 96
Appendix 5.E: Additional QoE questions in Post-trial Surveys 97
Appendix 5.F: Gameplay Survey Feedback 102

List Of Figures

3. Development

Figure 3.2.1: Early Mockup of Spell Combination Mechanic 20
Figure 3.3.1: Initial Sketch of the Map 21
Figure 3.3.2: Refined Map Design Document 22
Figure 3.3.3: Final Map with Annotations 23
Figure 3.4.1: Element Buttons Created using Online Resources and Software 24
Figure 3.4.2: A Character Model from the Game Infinity Blade (top) and a Character Model from the Game Paragon (bottom) 26
Figure 3.6.1: Blueprint Code for Spawning Players into the Match as their Selected Character 28
Figure 3.8.1: Character Hitboxes without Compensation 32
Figure 3.8.2: Character Hitboxes with Compensation at 100 Milliseconds of Latency 32
Figure 3.8.3: Character Hitboxes with Compensation at 200 Milliseconds of Latency 32
Figure 3.9.1: An Abbreviated Diagram Illustrating the Latency Compensation Algorithm 36

5. Results

Figure 5.1.1: Gaming Skill with Mouse and Keyboard (n=25) 46
Figure 5.1.2: Skill with First-Person Shooter Games (n=25) 46
Figure 5.1.3: Rating of Previous Cloud Gaming Experience (n=7) 47
Figure 5.2.1: Distribution of Spells Cast 49
Figure 5.2.2: Cumulative Distribution of Fireballs Cast Across All Trials from Test Group (left) [n=126] and Across All Trials from Test Group After Data Removal (right) [n=95] 51
Figure 5.2.3: Cumulative Distribution of Fireball Accuracies Across All Trials from Test Group 52
Figure 5.2.4: Quantile-Quantile Plot of Fireball Accuracies Across All Trials from Test Group 52
Figure 5.2.5: Average Fireball Accuracy vs. Latency for Test Group 53
Figure 5.2.6: Average Fireball Accuracy for Test Group vs. Latency with Average Base Latency for 0ms 56
Figure 5.2.7: Cumulative Distribution of Lightning Strikes Cast Across All Trials from Test Group (left) [n=126] and Across All Trials from Test Group After Data Removal (right) [n=40] 57
Figure 5.2.8: Cumulative Distribution of Lightning Strike Accuracies Across All Trials for the Test Group 58
Figure 5.2.9: QQ Plot of Lightning Strike Accuracies Across All Trials for the Test Group 58

Figure 5.2.10: Lightning Strike Accuracy vs. Latency for Test Group 59
Figure 5.2.11: Average Lightning Strike Accuracy for Test Group vs. Latency with Average Base Latency for 0ms 62
Figure 5.3.1: Average QoE (Quality of Experience) with Compensation on vs. Compensation off for each added latency category (0,100, and 200). 64
Figure 5.3.2: Average QoE with Compensation On vs. Off 65
Figure 5.3.3: Average Perception of How Accurately the Game Reflected User Skill Across Trials 65
Figure 5.3.4: Average Perception of Fairness Across All Trials with Compensation on v.s. Compensation off for Test Group 66
Figure 5.4.1: How Well Do You Believe the Bots Successfully Imitated an Average Human Player? (n=25) 67
Figure 5.4.2: How Much Did the Map Improve the Gameplay Experience? (n=24) 68
Figure 5.4.3" How Clear was the Objective of the Game? (n=25) 68
Figure 5.4.4: How Did You Feel about the Number of Spells? (n=24) 69

List of Tables

4. Evaluation

Table 4.2.1: Network Latency to Stadia Servers 39

5. Results

Table 5.1.1: Summary of Ages 43
Table 5.1.2: Summary of Majors 44
Table 5.1.3: Summary of Minors 44
Table 5.1.4: Weekly Time Spent on Video Games 45
Table 5.1.5: Preferred Gaming Platform 45
Table 5.1.6: Summary Table of Demographics Statistics 48
Table 5.2.1: T-Test Between Fireball Accuracies for with Compensation On and Off for the Test Group, Alpha = 0.10 54
Table 5.2.2: T-Test Between Non-Lagged Group's Average Fireball Accuracy and Each of the Test Group's Average Fireball Accuracies, Alpha = 0.10 55
Table 5.2.3: T-Test Between Lightning Strike Accuracies With Compensation On and Off, Alpha = 0.10 60
Table 5.2.4: T-Test Between Non-Lagged Group Average Lighting Strike Accuracy and Each of the Test Group's Average Fireball Accuracy, Alpha = 0.10 61
Table 5.4.1: Summary of Statistics 70

1. Introduction

Video games are extremely popular. Cloud gaming, a new way to play video games over the Internet, is becoming more accessible due to growth in commercial cloud gaming platforms [1]. Recent statistics have found that the worldwide market value for cloud gaming has increased from 2020 to 2021 [2].

Cloud gaming services allow users to play games on-demand over the Internet, while limiting hardware required by players [1]. Cloud gaming systems render the game scenes on cloud servers and stream the encoded scenes to "thin" clients over high capacity networks. User inputs such as from mice, keyboards, joysticks, and touchscreens, are transmitted from the thin clients back to the cloud servers. By using cloud hardware to run the games and then stream a video to the user while accepting their inputs, cloud gaming services can provide access to a wide audience on a variety of platforms, without users needing high-end hardware nor installing the games themselves.

An example of such a cloud gaming service is Google Stadia, owned and operated by Google, accessed through the Google Chrome browser, Chromecast, Android, and Safari on iOS [3]. As with all cloud gaming services, the user only requires a client with minimal hardware and software. In order to facilitate play on multiple platforms, Stadia has an original controller that can be used, but other input devices such as mouse and keyboard are also supported.

One major challenge facing cloud gaming systems is latency. Latency is the delay between a user's inputs and the corresponding, discernable responses from the game system. There are a number of factors that contribute to latency in cloud gaming systems, including playout delay (the time the client takes to render the game state and then process and send out the user's input), network delay (the time it takes for information to be sent to the cloud), and

processing delay (the time spent by the server to process the inputs and send back the corresponding visual changes relating to the game state) [4]. While network delay is often considered the primary factor when it comes to users experiencing latency in traditional network games, research suggests that a considerable amount of latency results from overload in the user's machine [5]. Fortunately, cloud gaming alleviates the client's required computational power by using the cloud service's hardware to do a majority of the processing. However, network delay is still of concern as the client cannot render game state based on inputs; it merely sends them to the server. This can create a feeling of unresponsiveness as each input must be sent to the cloud, processed by the server, and the video frames sent back to the user before they are seen.

When considering techniques to compensate for latency, a feeling of unresponsiveness and possible detriments to the quality of experience (QoE) should also be considered [6]. Studies have shown a relationship between latency, player performance, and QoE [7]. Specifically, as delay increases, performance decreases which in turn decreases QoE. Some games are more sensitive to the effects of latency, depending on the type of actions players carry out in-game [8].

To mitigate the effect of latency in games, various latency compensation techniques have been created. Among these methods, some seek to alter the game world, while others focus on providing some kind of assistance to players in response to latency [7, 9]. While these techniques vary in their approaches, Savery and Graham note that these algorithms each have to deal with the concept of time [10]. Since it takes time for messages to travel over a network, each client and server has a different frame of reference. This causes an increase in complexity, making it more difficult for game developers to implement them.

One type of latency compensation technique, known as attribute scaling, focuses on manipulating the properties of objects in the game world according to the amount of latency that is present. Examples of attribute scaling include altering the speed or size of objects. In a study run by Sabet et al., it was observed that attribute scaling was able to improve QoE in the presence of latency [7]. However, this same study also found that if the compensation technique is applied when no latency is present, players become bored because of the lower difficulty.

While there is some research regarding the efficacy of latency compensation on cloud gaming platforms, the aforementioned increase in popularity of cloud gaming demands more research to be done in this area. Furthermore, many of the studies concerning latency compensation focus on specific gaming actions in single player environments. While this provides important, fundamental insight into the nature of latency in games and how to compensate for it, there are also many games that are far more complex and include multiplayer environments. For those studies that do evaluate latency compensation in multiplayer games, it is typically done with a typical client-server structure. Finally, while there has been some research into attribute scaling, there are numerous types of attributes that can be scaled as well as games and network latency conditions to evaluate. To bridge this gap, our goal is to implement and evaluate an attribute scaling technique in a complete game on a cloud gaming platform.

We created an original, multiplayer game to test a novel attribute scaling latency compensation technique. We devised a method of manipulating the hitboxes of targets based on the latency a user experiences to make it easier for them to hit their target despite the adverse effects of latency. We developed the game in Unreal Engine 4, deployed it to Google Stadia, and conducted a user study to evaluate the effectiveness of this technique.

Our user study had 26 participants, who each played through 1 tutorial and 6 trial matches of our game, lasting three minutes each. There was one player and two AI bots on each team. One player had a random permutation of possible added latencies, either 0, 100, or 200 milliseconds, and latency compensation on or off. The other player had no added latency and no compensation for all trials. Of the 26, 18 were lagged players and 8 were non-lagged players. Players filled out a demographics survey before playing, post-trial surveys after each game, and a final survey about their overall gameplay experience. The surveys were analyzed for Quality of Experience (QoE) assessment.

We compared the average accuracy for two spells among lagged players, Fireball and Lightning Strike, because these were commonly used spells that players had to aim. There was a statistically significant improvement in accuracy with latency compensation at 100ms and 200ms of added latency, while the difference at 0 was not significant. For QoE questions, there was no statistically significant difference between trials with compensation on and off, but QoE did decrease at higher added latency values.

The remainder of this report is organized as follows; Chapter 2: Related Works describes previous work in this area of study and the limitations of previous research in regards to our specific subject area; Chapter 3: Development details the process of game development and employing our latency compensation technique; Chapter 4: Evaluation describes our user study to evaluate the effectiveness of our latency compensation technique in a cloud gaming architecture; Chapter 5: Results provides an analysis of our findings on our latency compensation technique and its effectiveness for cloud gaming; Chapter 6: Conclusion summarizes and concludes our findings; and Chapter 7: Future Work describes potential areas for study and research relating to our work.

2. Related Work

Past research into cloud gaming and latency compensation has examined various systems and latency compensation techniques, and their effects on user experience and performance. This section summarizes background information and past studies on latency and interactivity, latency compensation, cloud gaming, and the Google Stadia service. It also describes how these past papers relate to our project.

2.1 Latency and Interactivity

Latency, also known as lag by gamers, is the delay between when an action is taken and when the effect of the action is seen or heard by the user. For games, this means the delay between when a player executes an action and when the outcome of that action takes place on-screen. Two common types of latency are local latency and network latency. The former is related to the local machine that is running or streaming the game [11]. Local latency is more precisely defined as the sum of all latencies relating to the local machine from when a player presses a button to when their action is represented on-screen. Examples of sources contributing to local latency include the time it takes for the display to update properly, and the delay between when a key is pressed and when it is received by the application. As Chen et al. [3] point out, another example of local latency is when the computer's CPU/GPU is overloaded due to a lack of computational power. A major cause of latency for online multiplayer games is network latency, which is the time it takes for a packet to travel from the source to the destination and then back to the source [12]. Network latency on the Internet is typically longer and less consistent compared to local latency due to the variety of network conditions a PC may be subjected to.

According to Claypool et al. [13], some player actions are affected more by latency than others. These actions are made up of the following characteristics: precision, deadline, and impact. Precision is how accurate an action has to be in order to be successful, deadline is how long it takes for a certain action to be performed, and impact is how the player's actions affect the game world.

Since latency can affect a player's effectiveness in executing game actions, it can affect the player's Quality of Experience (QoE). One study conducted by Sabet et. al. [14] discussed this concept by conducting a study where users played three games with different basic actions, such as jumping, with varying amounts of latency. The study found that while players could adapt to predictable latency spikes in some games, increasing the QoE, other games were not able to be adapted to by players, even with consistent latency. Another study by Long and Gutwin analyzed how different input devices influenced target selection actions [9]. Participants played games with a predominant target-acquisition mechanic, where users selected static or moving targets using a mouse, controller, drawing tablet, and touchscreen while being subjected to different amounts of latency [9]. Overall the players "felt less capable, had less fun, attributed performance less internally, became more frustrated and felt the cursor was less responsive" when latency was applied. Another study by Claypool [15] supports the assertion that there is an inverse relationship between latency and QoE.

We focused our project on developing a multiplayer first-person shooter since most player actions such as moving around, dodging bullets, and shooting are characterized by their high precision, low deadline, and high impact. Since these actions are dependent on low latencies for players to perform well, we hypothesized that if players perform well with the latency compensation techniques we develop, there will be an improvement in QoE.

2.2 Latency Compensation

Due to the increase in popularity of online games, developers have found various solutions to deal with issues caused by latency. Latency limits the ability for real-time combat since it takes time to transmit each player's actions over the network [8]. This means that when a player sees another player's action, that action may have already started and even completed, giving players no time to react. Games such as *World of Warcraft* compromise by basing their combat on special actions rather than real-time movements. This allows them to delay the execution of these actions, giving them enough time to be transmitted over the network. Another compromise is making visual indicators purely cosmetic, as shown in an example given by Savery and Graham [10] for the simulation of a broken window: even though the shards of glass are flying through the air, they do not deal damage to the player.

Game developers have also created specific techniques to modify game conditions when latency is present. According to Lee et al. [16], geometric compensation refers to modifying certain aspects of the game scene to compensate for the effects of latency [16]. They ran a study in which the vertical spacing of the pillars in the game *Flappy Bird* was modified in the presence of latency, and the effect on the player error rate was measured. The study found that the error rate with latency was similar to when there was no latency present, implying that geometric compensation can be used to maintain player performance in the presence of latency. However, this study did not explore the efficiency of geometric compensation when dealing with more complex games, or how it affects the quality of experience for players.

Sabet et al. [17] experimented with the effects of geometric compensation on player performance. In this case, geometric compensation involved adjusting the speed and scale of an object based on the amount of delay present. While this technique proved successful (meaning it

increased player performance), the authors suggested that it should only be used in the presence of latency to avoid boring the player. A developer's goal when using geometric compensation should only be to restore the game's difficulty to a level similar to when there is no delay.

Other potential latency compensation techniques include aim assistance. One of the most common tasks carried out in computer usage, and more importantly, gaming, is mouse-based pointing. Being able to do so accurately and in a timely manner is important, yet difficult to do in an environment with latency. Bateman et al. [10] investigated the effectiveness of three aim-assistance techniques that could help compensate for latency. The techniques used were: sticky targets, target gravity, and acquisition feedback. "Sticky targets" is an aim assistance technique in which the mouse cursor is slowed while going over a target by an amount proportional to whatever latency is being experienced. The "target gravity" technique brings the cursor toward the target with the pull being proportional to the latency as well as the distance from the cursor to the target. Lastly, acquisition feedback provides users with additional feedback when near and hovering over a target in hopes that it will allow users to compensate for the latency themselves given additional time to react.

Both "sticky targets" and "target gravity" increased performance significantly in terms of time-to-target and reduction of errors [10]. However, acquisition feedback saw no increase in performance whatsoever. While these results indicate that the two techniques might be useful in compensating for latency in target acquisition, we must also consider the limitations such as the potential unintended effects of these techniques. Although not tested in this study, if there are multiple targets in the user's field of view and they try to pass through one target to hit another, their cursor may stick on an unintended target and disrupt the user experience. Target gravity could negatively impact the user experience in a similar manner.

One of the main challenges when dealing with latency is dealing with time. According to Savery and Graham [8], latency compensation algorithms in network games need to deal with both time and shared state. However, this is difficult because many programming languages are not equipped to deal with time, meaning game developers need to carefully consider the value of shared data and when the value was held. In their study, they propose a new programming model to deal with this issue called Timelines. With this model, variables that describe the game's state, which multiple clients must know about, are stored as a series of values (called a "timeline") in which time is used as the index to access the value of that variable at that time.

Compromises in games, as well as geometric compensation, aim assistance, "target gravity," and the timelines model, are examples of the various techniques available for dealing with latency in network games. Some of the studies described above indicated a positive effect on player performance when using attribute scaling. In our project, we tested the efficacy of attribute scaling, specifically the scaling of hitboxes relative to latency, in cloud games. We chose to focus on hitbox alteration to avoid visual discrepancies between the actions the players perform and what is displayed on-screen.

2.3 Cloud Gaming

In recent years, cloud gaming has grown in popularity. These commercial cloud gaming systems each follow the same general architecture. A thin client on the user's device sends their inputs to cloud servers, where those inputs are processed and the game logic determines the new game state [18]. The server's graphics processor then renders the game scene as video, which is then encoded and streamed back to the user's client. The client then decodes the video stream and

displays the results of the user's inputs with minimal delay. Almost all computation is done in the cloud rather than locally on the user's device.

There are multiple benefits of utilizing cloud gaming over traditional locally run games. Players are able to play new, graphically intensive games on less powerful devices since they only need to decode and display the video stream provided by the cloud. This also means that any games available on the cloud service could be played on any platform that can handle the video streams and send inputs to the cloud. For example, PC or console exclusive games could be played on a tablet or smartphone. Additionally, players do not need to download or store game data, as this is all done in the cloud and games can be accessed quickly and easily through dedicated servers.

There are also some drawbacks to cloud gaming services. Increased latency has been a common downside of cloud gaming. Player input must be sent to cloud servers, where it will be processed, and a video stream must be sent back [18]. This introduces additional delay that is not present in local game systems. Consistently streaming high-resolution gameplay requires that the user has a consistent, high-speed Internet connection, otherwise frame rates and video quality may decrease.

Previous studies have examined the effects of latency on performance and user experience with cloud gaming services. Claypool and Finkel conducted a study using two cloud systems, GamingAnywhere and OnLive [19]. They recorded user performance and subjective data via surveys for various amounts of artificial latency. It was concluded that cloud games are sensitive to latency since user performance decreased by up to 25% with every 100ms of added latency. In terms of QoE, there was a significant drop in rating for latency values above 100ms. In traditional game systems, third person and omnipresent games are generally less sensitive to

latency than first-person games. In a cloud gaming environment, these games were as sensitive to latency as first-person games. In general, latency is a bigger problem for cloud gaming services than for traditional network games.

Another study by Sabet et al. [20] examined specific characteristics of cloud games that are sensitive to delay. One study was performed to determine these characteristics, with participants brainstorming and deciding on nine characteristics relating to the categories of game design, game scoring, user interaction, and inputs. A second study was conducted to analyze these classifications. Participants watched videos of gameplay from various games, rating each video and each of the nine characteristics. A decision tree was created with an accuracy of 90% on the test set, demonstrating that type of input, temporal accuracy, predictability, and input directions had the greatest effect on QoE.

We focused on implementing hitbox scaling in a cloud-based environment due to the aforementioned popularity of the technology. Since latency is a problem in cloud gaming systems, we tested whether this latency compensation technique could help maintain player performance despite increased latency due to the cloud gaming architecture. With the increase in the number of cloud-gaming platforms and users, it is important to address the technology's shortcomings and ensure that it can provide an engaging user experience in a variety of network conditions.

2.4 Google Stadia

Google Stadia is a cloud gaming service, released in 2019, that we used to deploy and test our game. Currently, Stadia employs a suite of latency compensation techniques including "negative latency" [23]. These techniques focus on reducing the amount of delay in the response

to user input and ensuring that input is received and relayed correctly. The first aspect of the negative latency concept is the prediction of future states combined with error compensation for mispredictions. The second aspect is dropping intermediate frames when an error is detected, effectively acting as “course correction” when the server prediction and user input drift apart. This can mean predicting future inputs, several frames ahead of the actual user input, and then sends them to the client, creating the illusion of instant response to user input. This technique can have difficulty processing inputs that occur quickly and are hard to predict, such as a user potentially attacking every frame. Cloud platforms attempt to address this with supersampling. By rendering the game at a much higher framerate on the Stadia hardware, they are able to receive more than one input per frame which subsequently makes it much easier to predict further inputs as well.

While these techniques increase the general responsiveness of games on the platform, each game requires its own fine-tuning. We tested a specific latency compensation technique, hitbox manipulation, with Stadia’s API and added latency to understand if it would provide substantial gains for our specific game and potentially other similar games.

3. Development

In order to evaluate the efficacy of latency compensation on a cloud gaming platform, our team developed a video game. By creating a video game of our own we were able to control key variables and modify the game for our specific needs when performing a detailed analysis of the latency compensation technique. This section describes the development of our game, the technical aspects involved with the process, as well as the implementation of our latency compensation technique.

3.1 Game Concept and Treatment

As stated previously, player actions with high precision, low deadline, and high impact are most sensitive to latency [13]. Keeping this in mind, we chose to develop a game within the first-person shooter genre due to its abundance of actions with the aforementioned qualities. However, as a unique twist, we developed a first-person shooter game that involves using spells rather than guns. This aspect of the game still allowed for player actions that are highly sensitive to latency, but that are not quite as mundane as shooting a gun. Due to the magical fantasy nature of our game concept, we settled upon the name “Catalyst.”

The concept of a game in the shooter genre that involves using spells was inspired heavily by a recent game known as Spellbreak by Proletariat, Inc. [25]. Additionally, the primary gameplay mechanic, Spell Combination, was inspired by a specific character in the game DOTA 2 known as Invoker [26]. By taking inspiration from these games and combining various interesting mechanics we were able to create a game that is both fast-paced and mechanically complex.

3.2 Primary Gameplay Mechanics and Rules

As mentioned previously, the primary gameplay mechanic in Catalyst is a mechanic we refer to as “Spell Combination”. Within the game, there are three elements that players have access to: fire, water, and lightning. By pressing one of three keys a player is able to “invoke” one of these elements. Invoking three elements in any order, with repetitions, players are able to create one of 10 spells within the game. An early physical mockup of this mechanic can be seen in Figure 3.2.1 below.

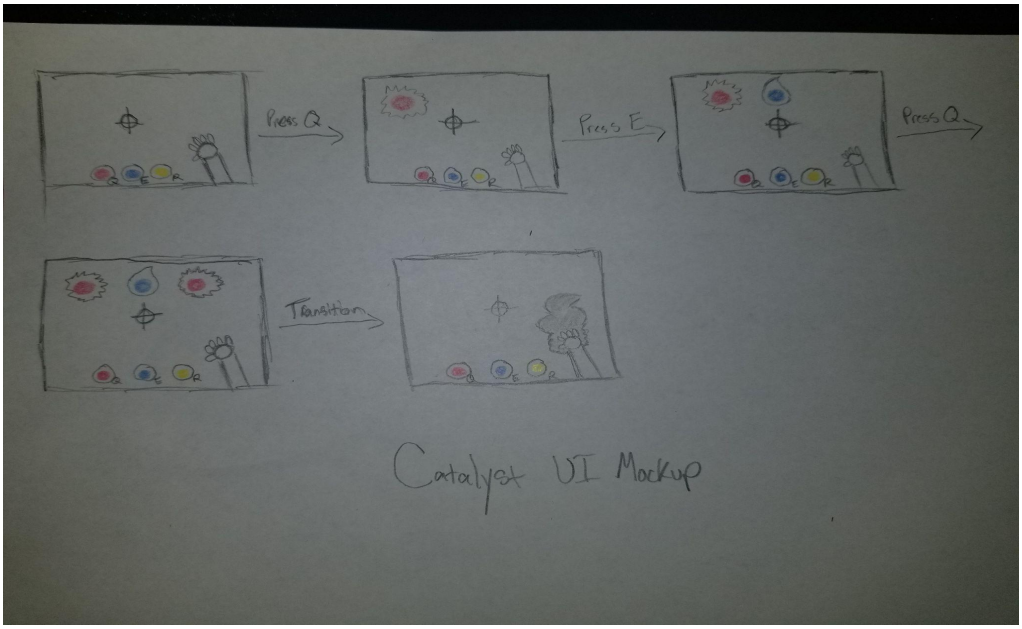


Figure 3.2.1: Early Mockup of Spell Combination Mechanic.

The Spell Combination mechanic provides players with quick access to a variety of tools for both attacking opponents and defending themselves and their allies. By having a number of latency-sensitive actions we were able to create a game that would provide a rich environment for studying the effects of latency compensation.

As previously mentioned, Catalyst is a multiplayer game in the first-person shooter genre. While designing an objective for Catalyst we considered objectives that would facilitate the full use of the Spell Combination mechanic and teamwork between players to showcase the full ability of the spells. To that end, we adopted a Capture the Flag game mode wherein two teams of three players face one another. We found three players on a team to strike a good balance between increased ability through teamwork and individual player skill expression.

3.3 Level Design

The primary objective of our game is to capture the enemy team's flag and return it to one's own base. However, a player with the enemy's flag is unable to successfully score while the enemy team is in possession of their own flag. This dynamic creates the need for combat and interaction between players. The level that the game is set in was designed with this game mode in mind and places each team at opposite ends of an arena. Initial level design documents can be seen in Figure 3.3.1 and Figure 3.3.2.

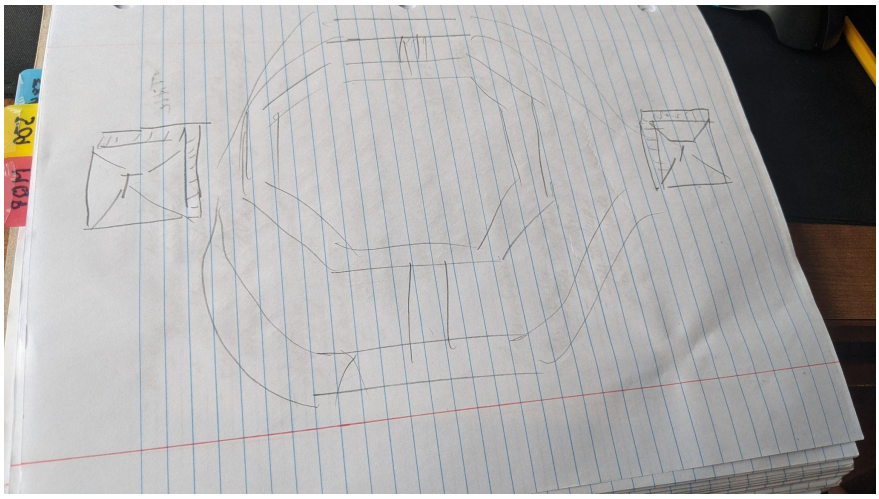


Figure 3.3.1: Initial Sketch of the Map.

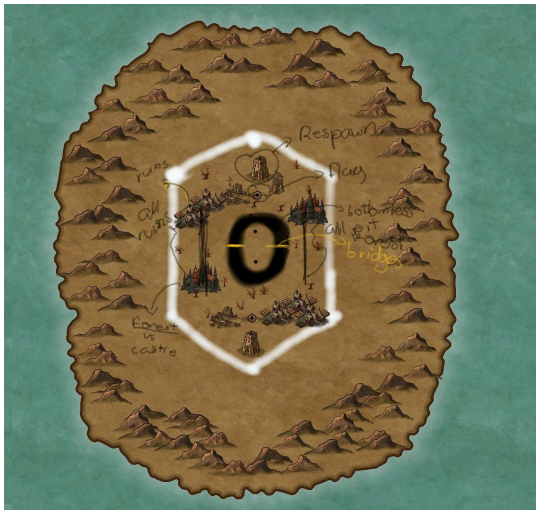


Figure 3.3.2: Refined Map Design Document.

The final aspect we considered when developing Catalyst was the inclusion of multiplayer. While having multiple players battle against one another is common within the shooter genre, it is also possible to create robust Artificial Intelligence (A.I.) actors as opponents for a single player. In addition, by having a multiplayer game we were better able to model a common game mode as well as study the effects of latency compensation on player perception of fairness.

These initial considerations and design decisions allowed us to create a game that suited our topic while also providing an interesting and rich environment for study and future work.

Figure 3.3.3: Final Map with Annotations.

The final design of the map can be seen in Figure 3.3.3 as it appears in the game. As noted in the design document each team has a small building that serves as a base and a respawn point. Additionally, there is a tower next to each spawn building that allows players to gain a vantage point when defending their flag. On either side of the map is a traversable area with obstacles that provide cover. On the left side of the map is an area with various fallen structures and ruins. On the right side of the map is an area with vegetation and small hills. The final notable feature of the map is the large central pit with only a small platform above it. Falling into the pit results in instant death and as a result players are forced to either battle upon the small platform or meet one another in the traversable areas on either side. By reducing the traversable area we sought to increase interplayer interaction.

3.4 Art

The Catalyst development team did not include a dedicated artist. As a result, all assets were either taken from free sources or made using basic tools such as the software paint.net [27].

For example, one of the main UI elements, the element buttons at the bottom of the screen required the use of free vector images from a website known as Vecteezy and an online program called Tokenstamp as shown in Figure 3.4.1 [28, 29].

Figure 3.4.1: Element Buttons Created Using Online Resources and Software

Much of the art was used or developed based on necessity and for the purpose of facilitating gameplay rather than for purely aesthetic purposes (Appendix 3.A). In other words, a specific visual art style was not at all a concern during the development of Catalyst. Because of this design paradigm, a majority of the art involved in the game remained in the same state throughout development. The one element that did see significant change, however, was the character models for player characters.

Initially, player character models consisted of only the base character model provided within Unreal Engine 4. However, after initial gameplay mechanics were completed, development became multiplayer-centric, and different character models became desirable if not necessary. In order to fulfill this need, we used a collection of free assets provided by Epic Games, the company that owns and develops Unreal Engine 4, from a game known as Infinity Blade [30]. By implementing eight models we were able to create a roster that allowed players to differentiate between six players while also improving the aesthetic value of the game.

The largest change in the art assets of Catalyst occurred after all core development goals were met. At this time our team discovered a collection of free character models from a failed game project by Epic Games known as Paragon [31]. We replaced the previous character models with the assets from Paragon due to their much higher quality and the various animations packaged with them. While this process significantly increased the file size of the game the increased graphical fidelity and expressiveness of the player character outweighed the cost of the increased size. Figure 3.4.2 provides a comparison between the Infinity Blade assets and the Paragon assets.

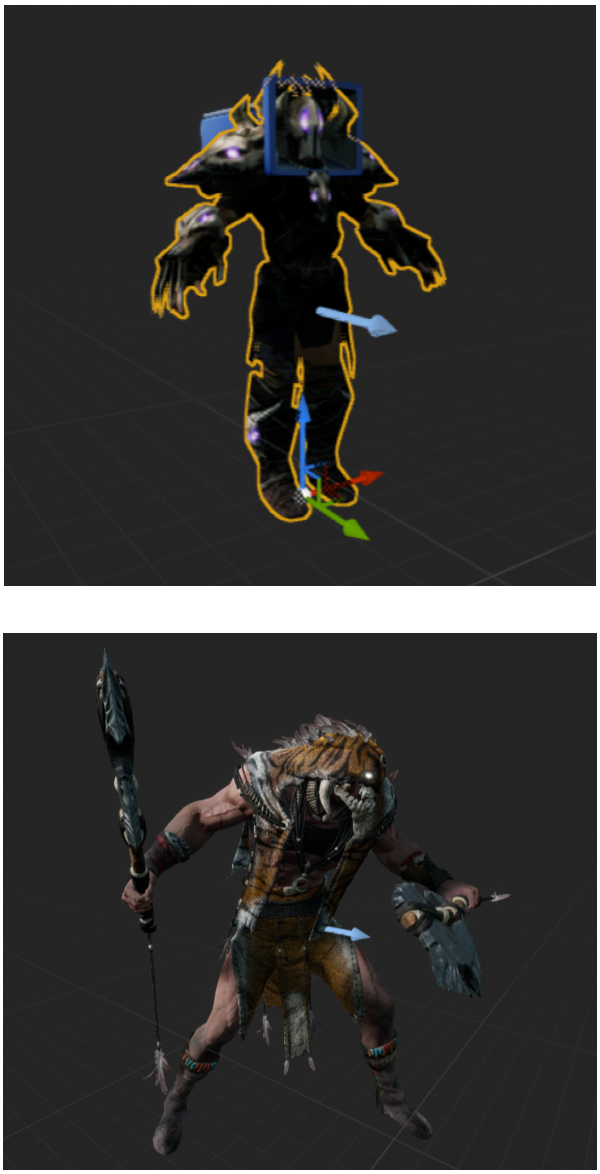


Figure 3.4.2: A Character Model from the Game Infinity Blade (top) and a Character Model from the game Paragon (bottom).

3.5 Sound Design

As with the visual art components, the Catalyst development team did not include a designated sound designer or artist. As a result, all sounds were obtained from collections of free sounds and music. Almost every action excluding movement has an associated sound or sounds that were chosen to best represent the aesthetic and mechanical essence of the spell. For example, the Mist Dispersion spell which involves teleporting a short distance and obscuring vision with a smoke cloud uses sounds that are intended to evoke a feeling of mystery and the unknown.

During the shift from Infinity Blade character models to Paragon character models, dialogue and other sound assets that were packaged with the character models were incorporated into the game for specific conditions. These voice lines were primarily used for the purpose of giving greater feedback to players of the state of their character. For example, a pain sound plays when a player takes damage which provides the player as well as the opponent that damaged them feedback that damage has been taken.

3.6 Technical Aspects

Catalyst was developed using Unreal Engine 4 Version 4.25 (and later updated to Version 4.26), a free game engine developed by Epic Games. We chose to use this particular game engine as one of our group members was familiar with the engine in addition to the existence of tools that would assist us in deploying our game to the cloud gaming platform Google Stadia for evaluation.

One of the unique features of Unreal Engine 4 is the ability to create gameplay code in two distinct ways. While it is possible to code a game using the engine's C++ functions, Unreal

Engine 4 also has a unique visual coding system known as “Blueprints”. These Blueprints are similar to other visual programming languages such as “Scratch” in that all operations are represented visually through the connection of graphical elements called “nodes”. However, the unique aspects of the Blueprints system are the large degree of flexibility provided by the numerous nodes and the ability to create new operations through functional programming or even C++ code.

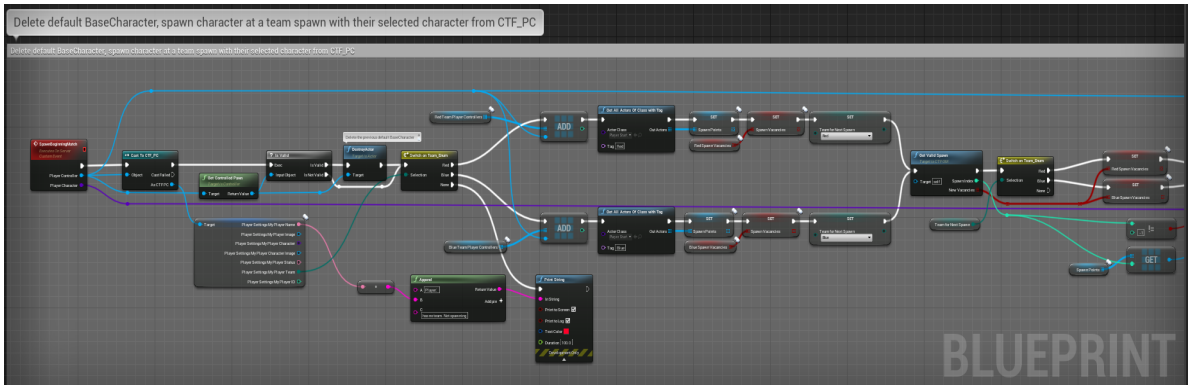


Figure 3.6.1: Blueprint Code for Spawning Players into the Match as their Selected Character

Figure 3.6.1 provides an example of a portion of code created using Blueprints. Colors and shapes are used to differentiate each node. For example, nodes that are mostly transparent with a slight coloring at the top represent objects and basic functions that manipulate objects such as getters and setters. The white lines that connect nodes represent the path of execution, these lines can be thought of as train tracks or roads with branches and switches serving the purpose of providing conditional execution. Finally, the colored lines pass object data into function nodes.

However, one of the significant downsides of programming with Blueprints is the numerous issues with version control. While the functional outcome of programming with Blueprints is the same as with programming using C++, the files generated for each Blueprint class are not text files, but rather binary files. What this ultimately means is that software such as Git does not identify the specific differences between two different versions of the same Blueprint file and thus cannot merge these two files effectively. This often defeats the purpose of version control using Git and forces a team of developers to perform version control manually. When developers fail to communicate, it is possible to end up with merge conflicts and the inability to bring changes to the same Blueprint class together. At times it may even be necessary to identify the changes one developer made to a class and manually recreate them in one version of the class. This can be a tedious process that deters developers, but with careful communication, the benefits of programming with Blueprints such as the easily visible visual code, the rapid iteration, and integration with the many visual elements of Unreal Engine 4 can be realized.

One of the other significant technical challenges during the development of Catalyst came from networking while creating a multiplayer game. A significant portion of these issues arose from a lack of understanding of network architecture in games and specifically in Unreal Engine 4. For example, understanding that client instances (those that connect to the main authoritative server) do not maintain any game state. As a result, when events occur the server must send a trigger that causes that event to run on the client. This can often result in events occurring twice as the client attempts to update the game state when instead it should rely upon the server replicating state to it. As our team gained familiarity with Unreal Engine 4 and the architecture of network games fewer mistakes were made and errors were more easily corrected.

3.7 Alpha Testing

Two of the most important aspects of the game development process are testing and iteration. In order to facilitate this process, WPI holds an event in the fall known as “Alphafest” which allows student developers to display early versions of their games and solicit feedback from a pool of users. Our team displayed Catalyst at this event and the feedback helped shape the direction of our development.

While a significant portion of the feedback we received was positive, indicating the overall enjoyability of Catalyst at that stage, there was also feedback that allowed us to improve the game. For example, much of the constructive criticism centered around the learning curve of the game and the difficulty in memorizing the spell combinations, of which there are ten. In order to mitigate this issue, we incorporated a sidebar that allows the player to quickly reference spell combinations and the effects of certain spells. By taking this into account, as well as other feedback surrounding things like the UI elements, we were able to improve Catalyst as a game experience.

3.8 IQP User Study

During the beginning of this project, we worked closely with a team of junior WPI students working on their Interactive Qualifying Project that covered a closely related topic. The focus of the IQP study was the creation of a model for the necessary values to compensate for latency using attribute scaling techniques [23]. In order to facilitate this our team created a version of Catalyst that was extremely limited in scope, this allowed the analysis of the effects of latency and our novel latency compensation technique on player performance.

After conducting a thorough user study and analyzing the data they collected, the IQP team was able to provide us with a formula for our latency compensation technique based on the following values: Accuracy, Latency, Scale, and Difficulty.

Accuracy represents the intended player accuracy when performing the action the team studied, firing a fireball spell, which is a small projectile, at an enemy as a float value from 0 to 1. Latency represents the latency that the player is experiencing currently measured in milliseconds. Scale is the float value factor by which we increase the sizes of the enemy hitbox. Finally, Difficulty is a measure of enemy speed, which plays a significant role in a player's ability to hit their opponent measured in cm/s. The following is the final equation the IQP team created by analyzing their data:

$$Accuracy = -0.72114883 * Latency + 0.21163254 * Scale - 0.00097065 * Difficulty + 0.67993323$$

By isolating the Scale variable we were able to obtain a formula for a scale factor at a given latency and difficulty to achieve a specific accuracy. The formula is as follows:

$$Scale = (0.72114883 * Latency + 0.00097065 * Difficulty + Accuracy - 0.67993323) \div 0.21163254$$

However, during development of our latency compensation technique and the preliminary/informal testing thereof we found that the scale factor had too little of an effect on the hitbox size as latency increased. This can likely be attributed to the differing conditions of the user study conducted by the IQP team compared to the full game. Whereas the user study involved a target at a fixed distance (within a range) the full game of Catalyst is a full 3D environment and thus a target could be at a variety of distances from the caster. In order to compensate for this effect we exaggerated the effect that latency had upon the scale value by increasing the coefficient for latency. We chose to double this coefficient as it exaggerated the size of hitboxes while not going outside of a reasonable range visually or in terms of the increase

in accuracy we observed in informal testing. The final equation we arrived at for determining scale factor at a given latency is as follows:

$$Scale = (1.44229766 * Latency + 0.00097065 * 300 + 0.9 - 0.6799332274674559) \div 0.21163254$$

Figure 3.8.1: Character Hitboxes without Compensation

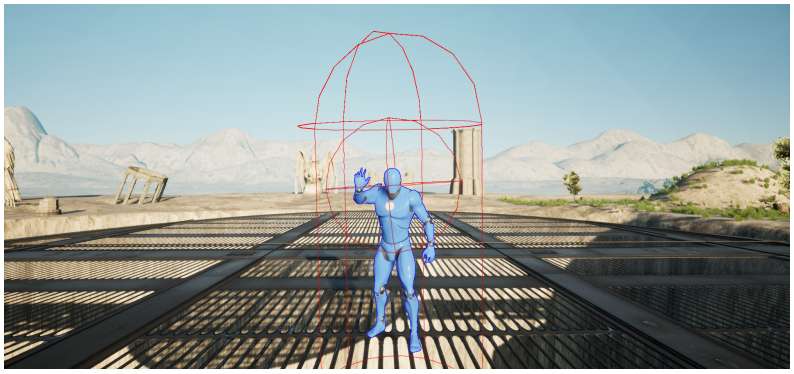


Figure 3.8.2: Character Hitboxes with Compensation at 100 Milliseconds of Latency

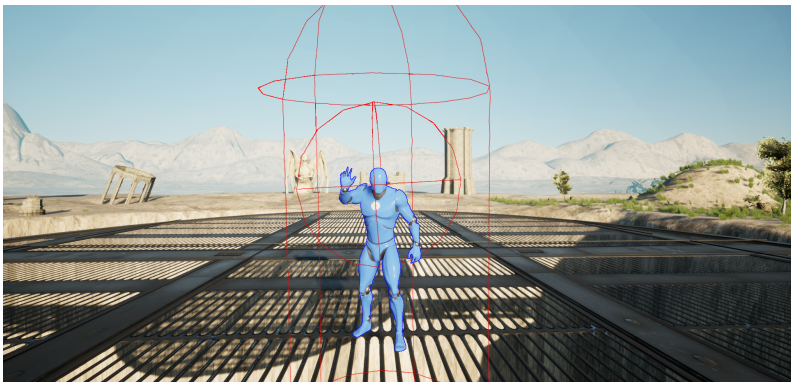


Figure 3.8.3: Character Hitboxes with Compensation at 200 Milliseconds of Latency

The value for accuracy was chosen to *over*compensate by scaling to attempt a 90% accuracy target for every player. While the IQP user study found that without compensation

player's demonstrated about a 68% accuracy on average we hoped to account for the likely non-linear nature of the detrimental effects of latency on player performance. The value for difficulty was chosen based on the movement speed of characters within Catalyst.

We then incorporated this formula into our game to create the implementation for latency compensation explained in the following section.

3.9 Latency Compensation Technique/Implementation

As stated previously, our latency compensation technique falls within the space of attribute scaling. Given this space we were left with the decision of what aspect of the game world to alter in order to improve player performance even with high latency conditions. In the case of Catalyst, we chose the performance metric of accuracy which naturally implied changing either the size of the target or the size of the weapon. Ultimately, we decided that altering the size of the target, in this case the enemy hitboxes, was more reasonable as it would be less noticeable to the players and thus have less impact upon their perception of the game's difficulty and fairness.

The basic premise of our novel latency compensation technique is increasing the hitbox size of enemy characters based upon the latency that a given player is experiencing. However, there are a number of elements we had to consider when implementing this technique.

First of these considerations was the nature of the client-server architecture. Possibly the simplest implementation idea for our technique would be simply having each player (client) have their own state associated with each enemy player's hitbox. This would allow the adjudication of a hit to occur within the client and it would not conflict with the different latencies that other players on the team are experiencing. However, this fundamentally contradicts the authoritative

server architecture that exists within Unreal Engine 4 in its base state. So in order to maintain the advantages of the authoritative server model and avoid additional conflicts with the replication of game state we held all game state information within the server and considered all players' latencies simultaneously.

Secondly, when attempting to hold all game state within the server we had to consider how to scale hitboxes correctly and when to scale them. This aspect of our implementation involved much thought, deliberation, and debate. Ultimately, our team decided that the best way to handle this was with what we call "dynamic" scaling. This involves scaling hitboxes at the time of hit detection in order to properly adjudicate hits. This remains in line with the authoritative server model as the client holds no unique state and also does not determine any interactions within the game world.

Finally, a common consideration with all attribute scaling techniques is the decision of how often the world should be altered to remain up to date with the player's changing latency. If the world is altered too infrequently the player is more likely to find themselves in a state where the world is not compensating for the latency they are experiencing and thus their performance suffers. Inversely, if the world is altered too frequently the player is likely to notice that the world is changing and also be unable to adapt their behavior. During development we found that when observing latency data collected during matches that there was little variance within a ten second period. Additionally, we found through gameplay that intense periods of player action, usually battling, rarely extended past ten seconds. As a result, we chose to scale hitboxes with updated values every ten seconds.

Considering all of the above factors our final latency compensation technique is as follows:

1. Each player has a “base state” or “outer” hitbox which is determined based on the latency of the player with the highest latency on the enemy team. This scale value is updated every ten seconds.
2. When a damaging spell from an enemy player strikes the “outer” hitbox of a player one of two events occur:
    a. The damaging spell was cast by the player with the highest latency on the enemy team. In this case, a hit is registered as the hitbox is already in a state that properly compensates for that player’s latency.
    b. The damaging spell was cast by a player other than the player with the highest latency on the enemy team. In this case, the hitbox is scaled down to a value that is appropriate for the latency of the player that cast the spell.
3. If the second case occurs two more possibilities arise from that:
    a. The spell continues on its path and strikes the hitbox that is now appropriately scaled for the caster’s latency in which case a hit is registered and after a brief period (to account for edge cases) the hitbox is scaled back up to the “base state”.
    b. The spell continues and does not strike the scaled hitbox in which case the hitbox still scales back up to the “base state” after the 0.5 second period.

This process is shown visually in Figure 3.9.1 with diverging arrows representing the different scenarios.

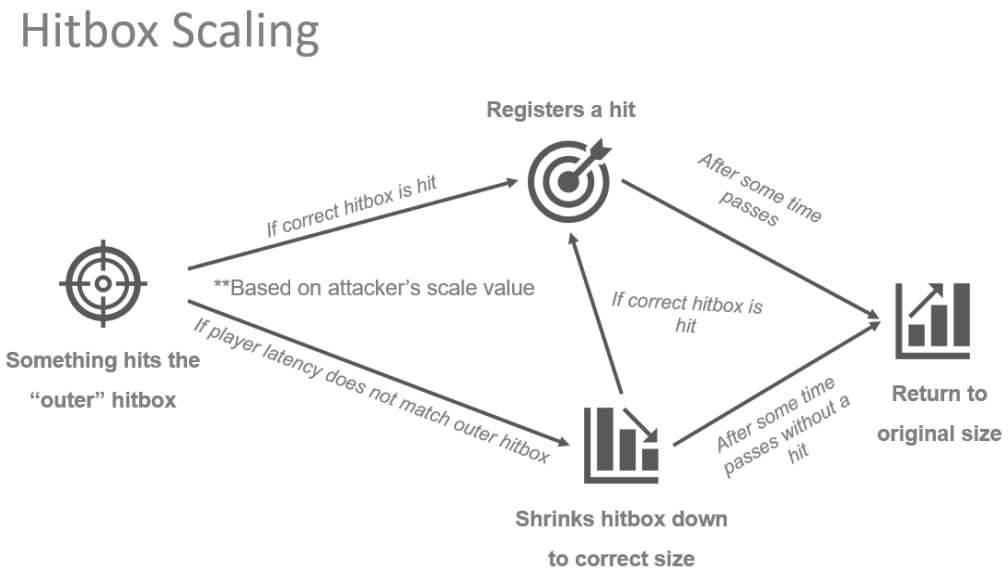


Figure 3.9.1: An Abbreviated Diagram Illustrating the Latency Compensation Algorithm

This technique meets all of the criteria by thoughtfully addressing all of the concerns when creating such an attribute scaling technique.

Firstly, the authoritative server model is maintained as previously mentioned, since the server determines the appropriate size for hitboxes and alters them which propagates to all clients. The server also handles all interactions and the adjudication of hits.

Second, by using "dynamic" scaling, the amount of compensation that is provided is adjusted for the specific player that is casting the spell. Our implementation also handles the edge case in which two players on the same team cast spells at the same target. By having a short delay of 0.5 seconds between a hit registering and the hitbox returning to the "base state" we avoid scaling back up while another player's spell is making contact. Additionally, a hit is adjudicated by assessing whether the latency of the player that cast the spell is equal to or *less than* the latency that the hitbox is currently scaled based on. This means that if the hitbox is

already scaled down to accommodate another player and a spell from another player hits in rapid succession, that player will still register a hit rather than having their spell disregarded.

Finally, we engage in “passive” scaling (changing the “base state” hitbox to match the most up to date latency value) in a 10 second period that is neither too infrequent nor too frequent which avoids common issues with attribute scaling techniques to the best of our ability.

4. Evaluation

Our main method of evaluation was conducting a user study where we recruited users to play multiple rounds of Catalyst. After each trial we collected subjective data and feedback, and also logged important player actions throughout all trials to determine accuracy. By collecting this data, we intended to determine how effective hitbox scaling is as a latency compensation technique.

4.1 User Study

Building a game from scratch allowed us to easily modify conditions and gameplay for a better user study experience. To this end, we created a separate version of the game specifically for the user study. We made various changes in this version, including removing the deathmatch event if the score was tied at the end of the game, automatically returning to the lobby after trials, and randomly generating a trial order to pass to a batch script, which is used to vary added latency.

4.2 Setup

The study took place in the Zoo Lab, a computer lab in the basement of the WPI Fuller Labs building. Each session consisted of one or two participants, with two AI bots on either team with them. If only one participant was scheduled for a time slot, one of the two proctors would play on the other team, with their data being discarded. Participants used laptops with the following specifications:

Processor: Intel(R) Core(TM) i7-8665U CPU @ 1.90GHz 2.11 GHz

RAM: 16.0 GB (15.8 GB usable)

GPU: Intel HD Graphics 620

Monitor resolution: 1920 x 1080 pixels

Screen Size: 13.3 in

Refresh rate: 60 Hz

Mouse: Dell HS8141611NF

OS: Windows 10 Enterprise 20H2 19042.928

Browser Version: Chrome 90.0.4430.93 (Official Build) (64-bit)

Ethernet Card: Intel(R) Ethernet Connection (6) I219-LM

The laptops were also connected over Ethernet (using a CAT-6 cable) versus WiFi to reduce network jitter and base latency. Additionally, all laptops used the same mouse and keyboard, making any local latency identical between laptops. Headphones were also provided to the users.

The local latency of each laptop was 39.25 ± 6.85 milliseconds. This measurement was taken using a high-speed camera filming at 1000 frames-per-second. We determined the time taken to open a context menu in Google Chrome to measure the time between clicking the mouse and the screen updating. In order to measure the latency, both the mouse and the laptop were in-frame and we counted the number of frames for the duration then dividing it by the frame rate.

Added Latency	Mean	Standard Deviation
0	11.0	2.3
100	114.0	2.9
200	214.3	3.2

Table 4.2.1: Network Latency to Stadia Servers

Table 4.2.1 displays the mean and standard deviation of total network latency in milliseconds at each added latency value for all users and trials.

When the host of the game first joins the lobby, the game generates a random permutation of trials and outputs this to a text file. Using a batch script, this file is retrieved and parsed, ensuring the correct order of added latencies is set by the Clumsy program [32]. Clumsy is a third party network tool that allows for various functionality, including lag, packet loss, throttling, and packet duplication. For this project we added latency to outbound packets. After each trial when the participant moved to a different lab computer to answer their survey questions, the proctors advanced the trials discreetly by pressing a key in the command window, starting the next latency value.

At the beginning of the study, the participant read through and agreed to the informed consent agreement and the COVID protocol. Next, they filled out a demographics survey to provide background information about themselves and their experience playing video games. Each participant played one practice round to learn the mechanics of the game, and then six trials. These trials had added latency values of 0ms, 100ms, and 200ms, with latency compensation on or off, resulting in a total of six trials. In each session, the player hosting the game had added latency, while the other player had no added latency and no latency compensation. After each trial, participants filled out a post-trial survey on a separate desktop with a few questions rating their experience during that game. All survey questions can be found in Appendices 4.A-4.C. After all trials were complete, participants filled out a final survey about their overall gameplay experience, and then were debriefed on the purpose of the study.

Throughout the study, appropriate COVID-19 precautions were taken to ensure the safety of the proctors and participants including social distancing, sanitization of all computers before

and after finishing a session, and applying disposable sanitary covers for the participants' headphones. We developed an AI player so we would only need two participants at a time (one on each team). Between each session, all participants and proctors were provided hand sanitizer, and participants and proctors kept 6 feet apart from one another at all times. This was done by placing the survey and gaming computers on opposite ends of the room allowing the proctors to move toward the gaming computers while maintaining social distancing.

4.3 Recruitment

To recruit participants for the user study, we advertised it on a variety of communication channels to Worcester Polytechnic Institute (WPI) students. We wrote an advertisement for the study explaining what the user would be doing and how long the session would take. This was accompanied by a video trailer for the game (Appendix 4.D). This advertisement was sent via email to students in the Interactive Media and Game Development (IMGD) and Computer Science departments at WPI. Additionally, the advertisement was posted in various WPI discord channels and the WPI subreddit. The study was also added to the WPI SONA Systems platform, where user studies can be posted for Psychology students to participate. We also provided academic credit for students at WPI who were taking IMGD or Psychology courses, as these students need to participate in playtesting and user studies as part of course requirements.

4.4 Data Collection

The game polls players every second for their current value of latency (as returned by the Stadia API), and logs these values to a log file. To collect objective data related to player performance, the game logs nearly all player actions, for each player. This includes actions such

as a hitbox being scaled, casting a spell, when a spell hits something (a player, wall, etc.), when a flag is picked up or dropped, and when a point is scored. At the end of each match, final stats for each player are logged with statistics such as number of kills and deaths. Each event is logged along with the appropriate information. For example, when a player's hitbox is scaled, the new scale factors for the head and body hitboxes are logged alongside the username of the player and the corresponding latency value used in the scale factor calculation. Each event was logged with a timestamp and contained delimiters for simple parsing. A sample of a log file can be found below in Appendix 4.E.

After each session, the log files were retrieved from the appropriate Stadia instances. These log files were passed into a Python script which parsed the log file, assembled a dictionary of statistics for each player, calculated player accuracy for each spell, and wrote all this information to a Microsoft Excel file. This excel file contained multiple sheets, one for each unique type of event that was logged. Once all trials were completed, all Excel files were placed into a single folder, where Excel could import and combine all of them into a single file. This file was then used for data analysis.

For subjective data collection, participants filled out a variety of surveys. Before participating in any trials, participants filled out a demographics survey that asked them questions regarding their previous experience with gaming, specifically on cloud gaming platforms (Appendix 4.A). After each trial, participants filled out a short survey regarding the quality of their experience during that trial (Appendix 4.B). Once all the trials were complete, participants filled out a final survey, which asked for their input regarding the design of the game itself (Appendix 4.C).

5. Results

This section discusses user data relating to four main categories: the demographics of the participants, player performance with regards to accuracy versus latency with hitbox scaling on and off, and user responses regarding Quality of Experience (QoE) as well as general feedback for Catalyst.

5.1 Demographics

We had twenty-six users participate in our study. Of this group, eighteen had varying latency and compensation conditions, and eight had no latency applied and no compensation throughout all trials. Twenty of the participants were male, and six were female.

Age	Number of Participants
18	1
19	2
20	7
21	4
22	9
23	1
25	1
31	1

Table 5.1.1: Summary of Ages

Table 5.1.1 displays the ages of all users who participated in our study. All users were between 18 and 31, with most users between 19 and 22. This was expected, since the user pool consisted of WPI students. Another demographic question asked the graduation year of the users,

which resulted in thirteen graduating in 2021, five in 2022, six in 2023, one in 2024 and one in 2025.

Major	Number of Participants
Computer Science (CS)	12
Interactive Media and Game Development (IMGD)	2
Robotics Engineering (RBE)	2
Mechanical Engineering	2
Aerospace Engineering	1
Biomedical Engineering	1
Environmental Engineering	1
CS/IMGD double major	3
CS/RBE double major	1
RBE/Electrical and Computer Engineering double major	1

Table 5.1.2: Summary of Majors

Minor	Number of Participants
No minor	16
Interactive Media and Game Development	3
Computer Science	2
Robotics Engineering	1
Concentration Manufacturing	1
Data Science	1
Mechanical Engineering	1

Table 5.1.3: Summary of Minors

Tables 5.1.2 and 5.1.3 show the majors and minors of the students who participated in the user study. More than half of the users had Computer Science as one of their majors, with five Interactive Media and Game Development and the remainder spread among various engineering disciplines. For minors, most participants did not have any, although there were a few CS and IMGD among others.

Average hours per week playing video games	Number of participants
0	3
Less than 1 hour	1
1-5 hours	9
6-10 hours	9
11-15 hours	2
More than 15 hours	2

Table 5.1.4: Weekly Time Spent on Video Games

Preferred gaming platform	Number of Participants
PC	14
Console	10
Mobile	2

Table 5.1.5: Preferred Gaming Platform

Table 5.1.4 summarizes the average amount of time each user spends per week playing video games. Most participants were in the 1-10 hour per week range. Table 5.1.5 shows the platform users prefer to play games on, between PC, console, and mobile devices.

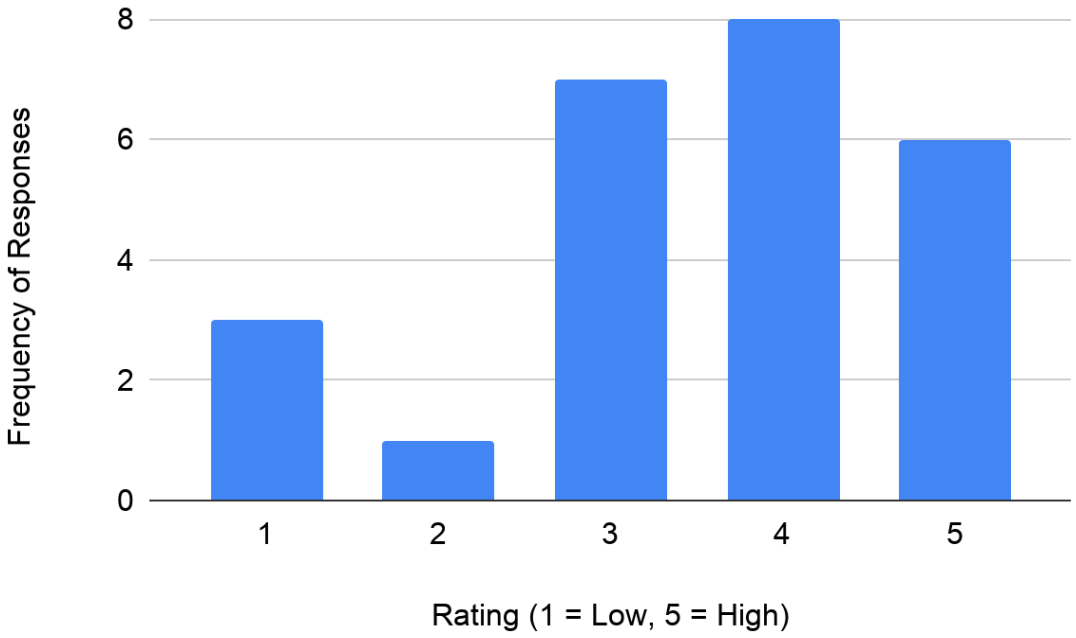


Figure 5.1.1: Gaming Skill with Mouse and Keyboard (n=25)

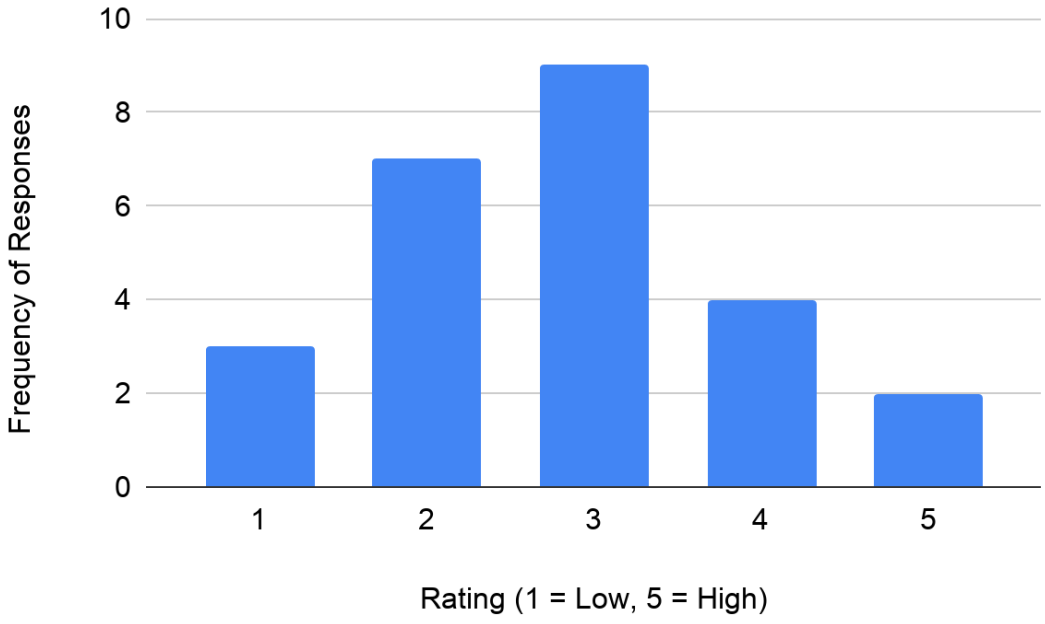


Figure 5.1.2: Skill with First-Person Shooter Games (n=25)

Figure 5.1.1 displays user's rating of their skill at playing games with mouse and keyboard. Figure 5.1.2 shows their self-rated skill rating playing First-Person Shooter (FPS) games. Both of these questions are relevant because Catalyst is an FPS game, and all players used keyboard and mouse to play the game. Since over half of the users preferred playing games on PC, it makes sense that most of them rated their gaming skill with mouse and keyboard on the high end, since most PC players use mouse and keyboard for input rather than a controller. The FPS skill graph is roughly a bell curve, with most users falling in the 2-4 range.

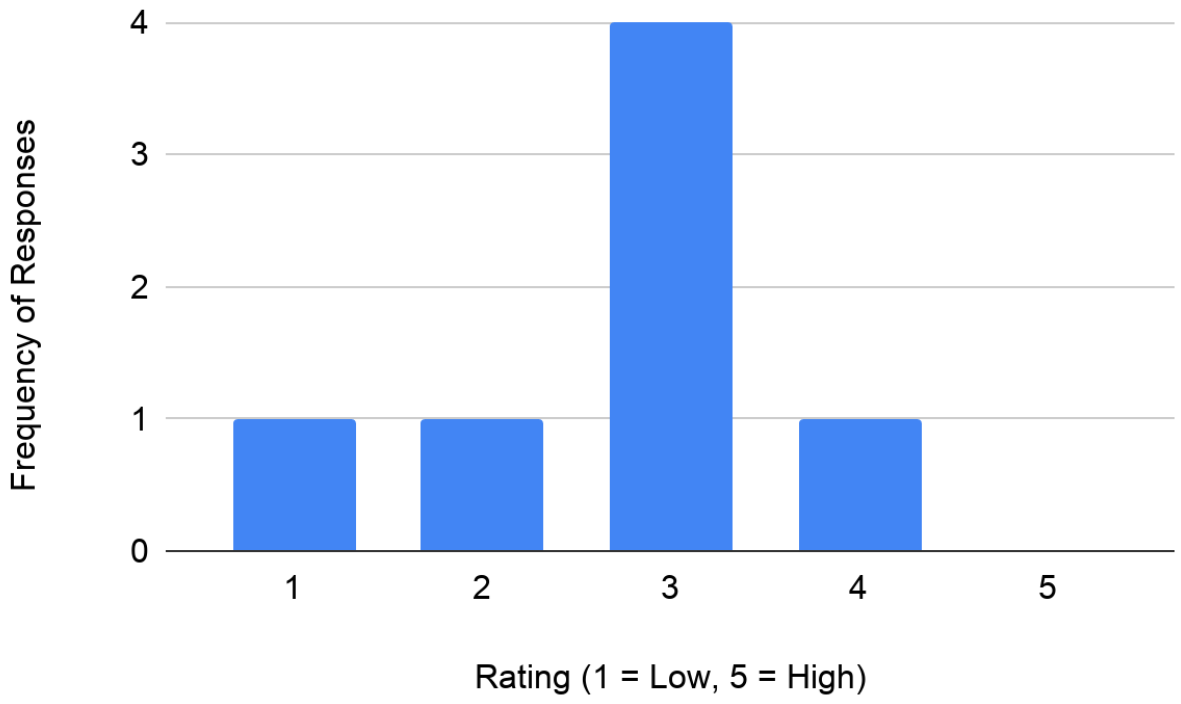


Figure 5.1.3: Rating of Previous Cloud Gaming Experience (n=7)

Nineteen users had no prior cloud gaming experience, six had used cloud gaming services, and one didn't know what cloud gaming was. Figure 5.1.3 shows a table of each user's rating of their cloud gaming experience, for those who had tried cloud gaming in the past. Out of the six who had tried cloud gaming, five had used Google Stadia.

Question	n	Mean	Standard Deviation
What is your age?	26	21.4	2.4
Rate your proficiency using mouse and keyboard for games.	25	3.5	1.3
Rate your skill with FPS (First-Person Shooter) games.	25	2.8	1.1
If you had experience with cloud gaming, rate your experience.	7	2.7	1.0

Table 5.1.6: Summary Table of Demographics Statistics

Table 5.1.6 summarizes the data for questions with numerical responses, displaying the number of responses, the mean, and the standard deviation for each of the 4 questions.

5.2 Player Performance

To analyze objective measures of player performance, we analyzed the accuracy of the spells that were used most often throughout the study. In each case, we only looked at data for the six 3-minute trials, not the tutorial match.

5.2.1 Distribution of Spells Cast

With eleven different spells (the default Arcane Impulse plus the 10 spell combinations), we analyzed the distribution of spells cast to see which were most often used. This allowed us to determine which spells we should analyze to determine player performance. The Pareto chart in Figure 4.2.1 details this distribution. The horizontal axis contains the 11 different spells that are

available to players. The left vertical axis measures the numbers of times that spell was cast, while the right vertical axis measures the cumulative distribution of spells cast.

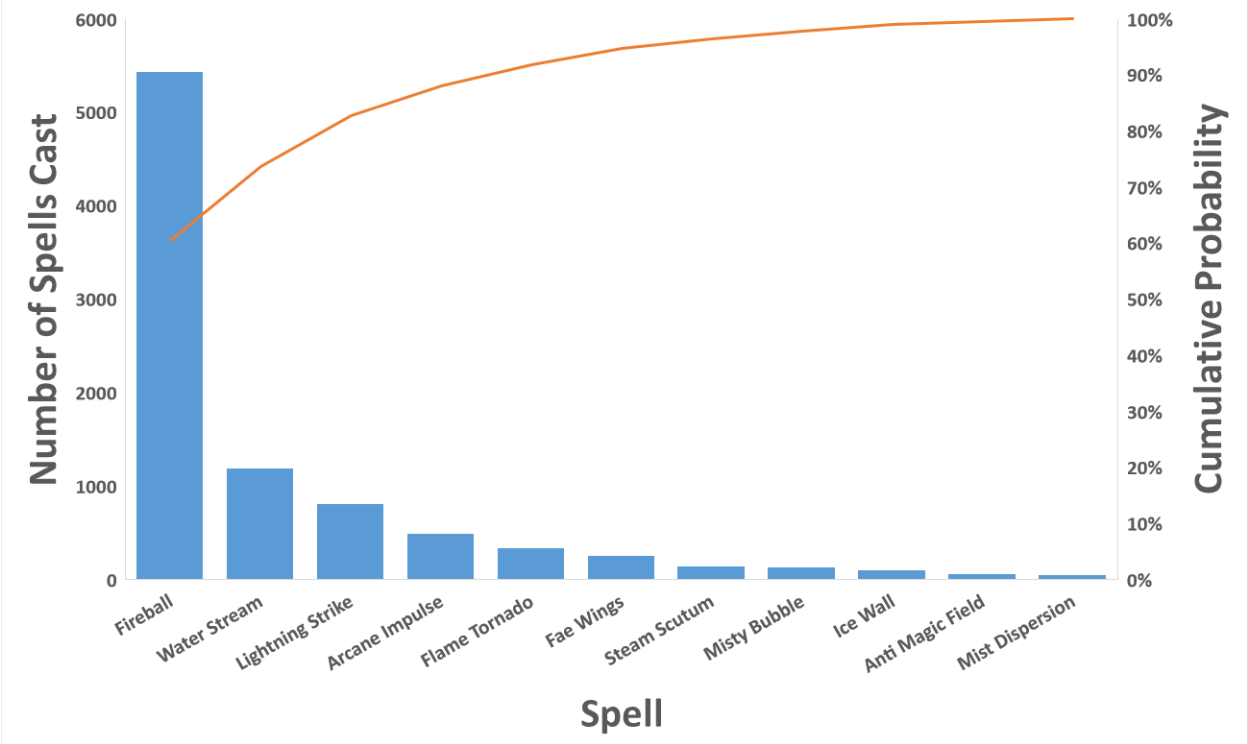


Figure 5.2.1: Distribution of Spells Cast

According to this figure, Fireball was the most popular spell, making up approximately sixty percent of spells cast. While the Water Stream spell was the second most popular, the spell also logged as a hit for every second the spell dealt damage, and only logged when it dealt damage, meaning if the user held the mouse button down for ten seconds on a target, it would be logged ten times. It also entails holding a reticle over a target for an extended period of time, making it difficult to to analyze the spell with regards to traditional measures of accuracy. The third and fourth most popular spells were Lightning Strike and Arcane Impulse. Fireball and Arcane Impulse are the only two projectile-based spells, while Lighting Strike is a hit-scan spell, meaning it draws a line trace in-game to determine a hit rather than spawning a projectile.

However, Arcane Impulse cannot be selected once the user switches to another spell; they only gain access to it again once they die and respawn. Therefore, Fireball and Lightning Strike were chosen for analysis with regards to player accuracy. These per-spell accuracies were calculated on a trial-by-trial basis. There are seven trials total: one tutorial match plus six trials. For this analysis, we ignored player accuracy during the tutorial matches, so we had six accuracies per participant, one for each trial.

5.2.2 Accuracy vs. Latency: Fireball

We first analyzed Fireball accuracy for the test group, which contained 18 participants for an initial total of 126 data points. We first removed data points for tutorial matches, since those were meant for the participant to become comfortable with the game. This removed 18 data points for a remaining total of 108. When analyzing this remaining data we noticed that many players who had 0 accuracy in a trial had cast none or a small amount of Fireballs during that trial. As a result, their accuracy of 0 was not representative of a player who was trying to hit an enemy with a spell under their current network conditions. We removed those data points for which no Fireballs were cast, which left us with 103 data points. For a non-zero number of Fireballs cast, we calculated a cut-off point. If a player had an accuracy of 0 and cast a number of Fireballs less than that cut-off point, that data point was removed as well. To calculate this cut-off point we analyzed the number of non-zero Fireballs cast when a player had an accuracy of 0 and identified a lower bound for outliers. The standard equation of $mean \ + / - \ (1.5 \ * \ IQR)$ was not appropriate as it provided a negative lower bound. A cut-off with a value of 1.0 instead of 1.5 in the equation also yielded a negative lower bound. Using a value of 0.5 in place of 1.5 yielded a cut-off point of 5.125. Therefore, we also removed

data points where a player's accuracy was 0 and their number of Fireballs cast in that trial was less than or equal to 5. This resulted in an additional 8 data points being removed, leaving us with a total of 95 data points. The cumulative distribution functions before and after this data removal can be seen in Figure 5.2.2, in which the left tail after data removal was not heavily weighted by a large number of little to no Fireballs cast. The rest of Fireball accuracy analysis is done with the test group after data removal.

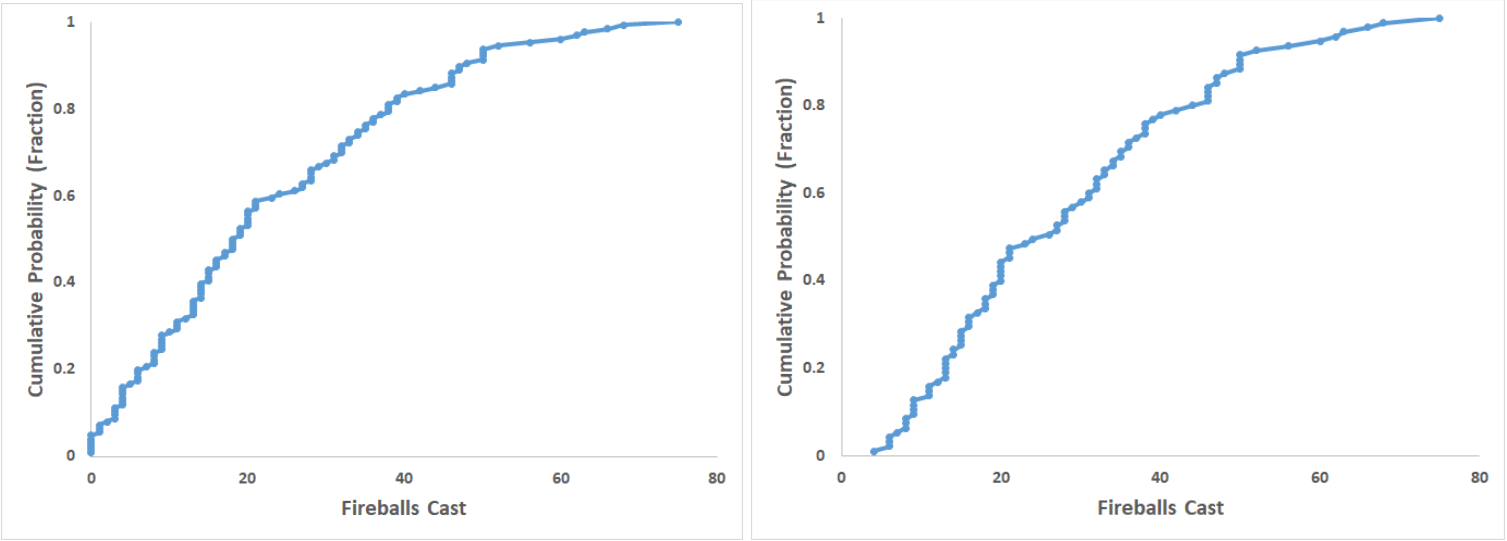


Figure 5.2.2: Cumulative Distribution of Fireballs Cast Across All Trials from Test Group (left) [n=126] and Across All Trials from Test Group After Data Removal (right) [n=95]

These figures indicate that the distribution of Fireballs cast was slightly normal. With these points removed, we analyzed Fireball accuracy. Figures 5.2.3 details the cumulative distribution of Fireballs accuracies across all network conditions. Figure 5.2.4 shows the Quantile-Quantile (QQ) plot between the distribution of all Fireball accuracies and a normal distribution.

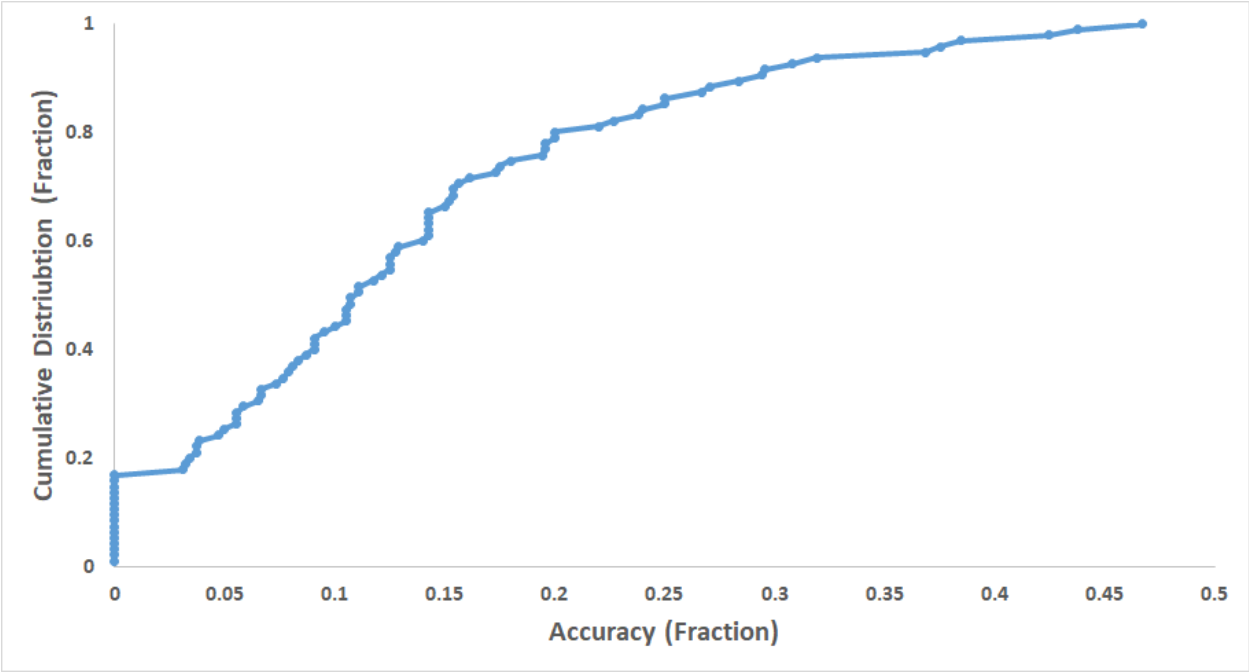


Figure 5.2.3: Cumulative Distribution of Fireball Accuracies Across All Trials from Test Group

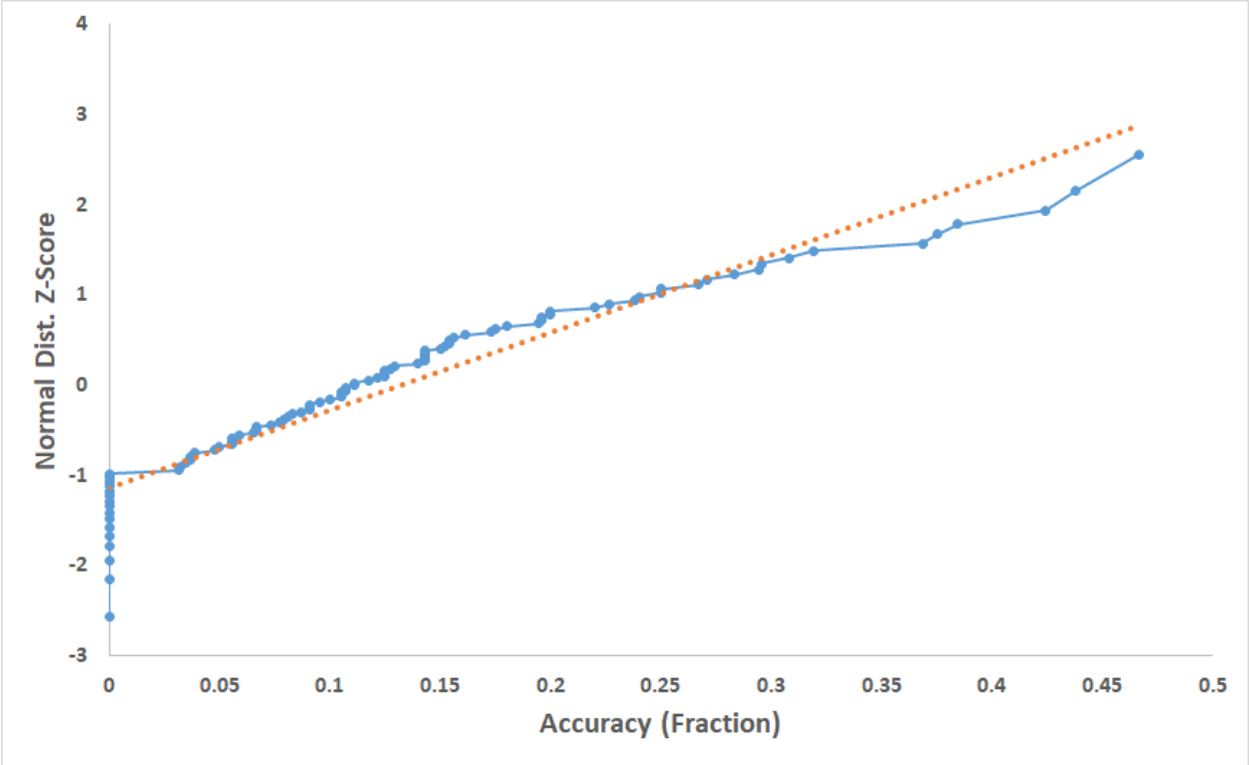


Figure 5.2.4: Quantile-Quantile Plot of Fireball Accuracies Across All Trials from Test Group

Other than the left tail in Figure 5.2.4, these figures suggest that the accuracies follow a slightly normal distribution. This matches the distribution of Fireball accuracies under each of the 6 individual sets of trial parameters (see Appendices 4.A and 4.B). For each value of added latency with compensation on and off, the QQ plots that are mostly linear.

With these distributions in mind, we calculated 90% confidence intervals for the average Fireball accuracy under each of the six different sets of trial parameters. This was done using a t-distribution. Figure 5.2.5 details the average Fireball accuracy versus latency, with hitbox scaling compensation on and off.

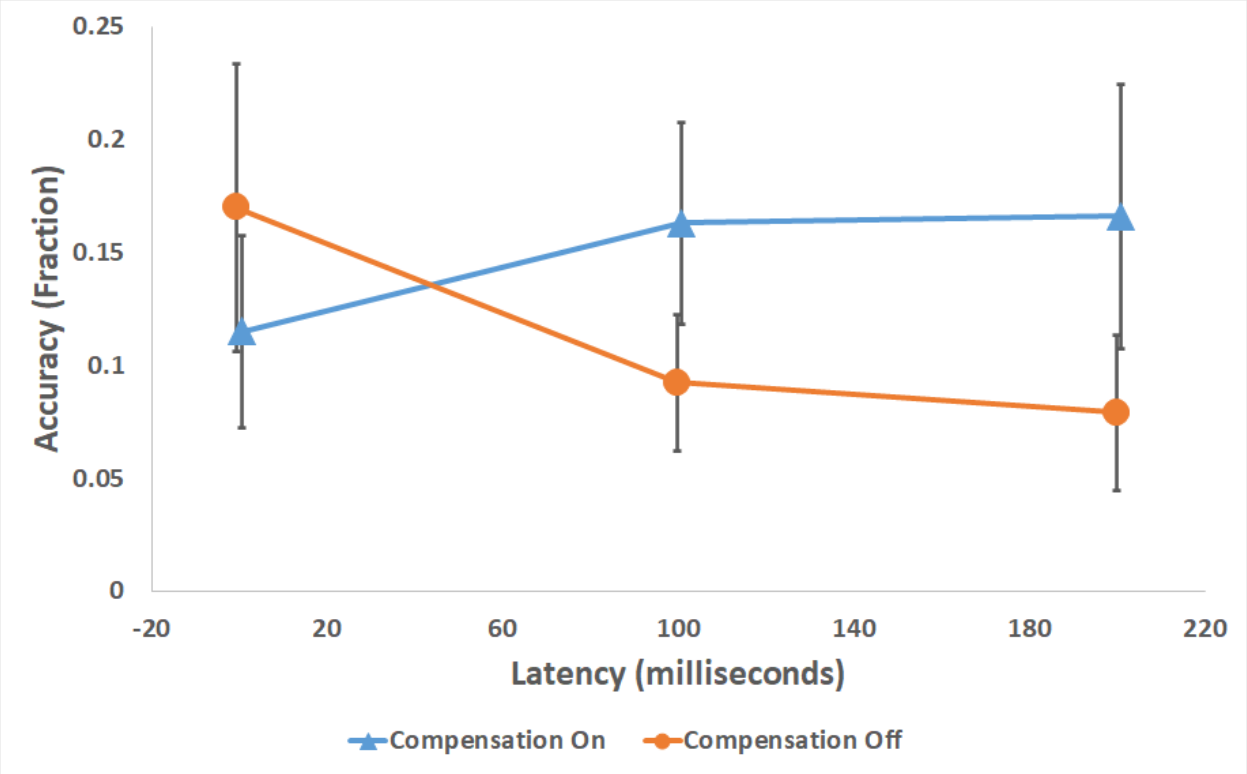


Figure 5.2.5: Average Fireball Accuracy vs. Latency for Test Group

To determine statistical significance of these results, we ran a t-test assuming unequal variance with an alpha value of 0.10. The results of the t-test can be seen in Table 5.2.1, where the p-values are one-tailed p-values.

Added Latency	Degrees of Freedom	T-Value	P-Value
0	25	-1.26	0.11
100	28	2.28	0.02
200	24	2.25	0.02

Table 5.2.1: T-Test Between Fireball Accuracies for with Compensation On and Off for the Test Group, Alpha = 0.10

The results indicate that there is no statistically significant difference between the average accuracies for 0ms, while there is a statistically significant difference between the accuracies for 100ms and for 200ms. As latency increased from 0ms, the average accuracy for the Fireball spell was higher with hitbox scaling on than with it off.

For comparison, we also analyzed the average Fireball accuracy of the 8 participants in the non-lagged group. Similar to the test group, we first removed data points where 0 Fireballs were cast. We then took the remaining data points that had a Fireball accuracy of 0 and calculated a cutoff point in a similar manner to what was done for the test group's cut-off point. In this case, the cut-off point was 6, so we removed more data points that had less than 6 Fireballs cast with 0 accuracy. This removed 6 more data points, leaving us with 37 data points for the non-lagged group where the average Fireball accuracy was 0.12 with a 90% confidence interval of [0.06, 0.18]. We then conducted t-tests with an alpha value of 0.10 between this average, which can be seen in Table 5.2.2. The p-values are two-tailed.

Added Latency	Compensation On?	Degrees of Freedom	T-Value	P-Value
0	Yes	44	0.13	0.90
0	No	33	-1.09	0.28
100	Yes	47	-1.14	0.26
100	No	52	0.83	0.41
200	Yes	36	-1.07	0.29
200	No	49	1.18	0.24

Table 5.2.2: T-Test Between Non-Lagged Group's Average Fireball Accuracy and Each of the Test Group's Average Fireball Accuracies, Alpha = 0.10

According to the table, none of the p-values were less than the chosen alpha value of 0.10, meaning that for each of the 6 trial conditions, there was no statistical significance between any of those average accuracies and the non-lagged group's average Fireball accuracy.

Since there was no statistically significant difference in Fireball accuracy for the test group at 0ms, we averaged together all the accuracies at 0ms from both the test and non-lagged groups (for a total of 67 data points) in order to find a "true" base accuracy when no added latency is presented. Figure 5.2.6 contains this true accuracy alongside the test group's accuracies for 100 and 200ms from Figure 5.2.5.

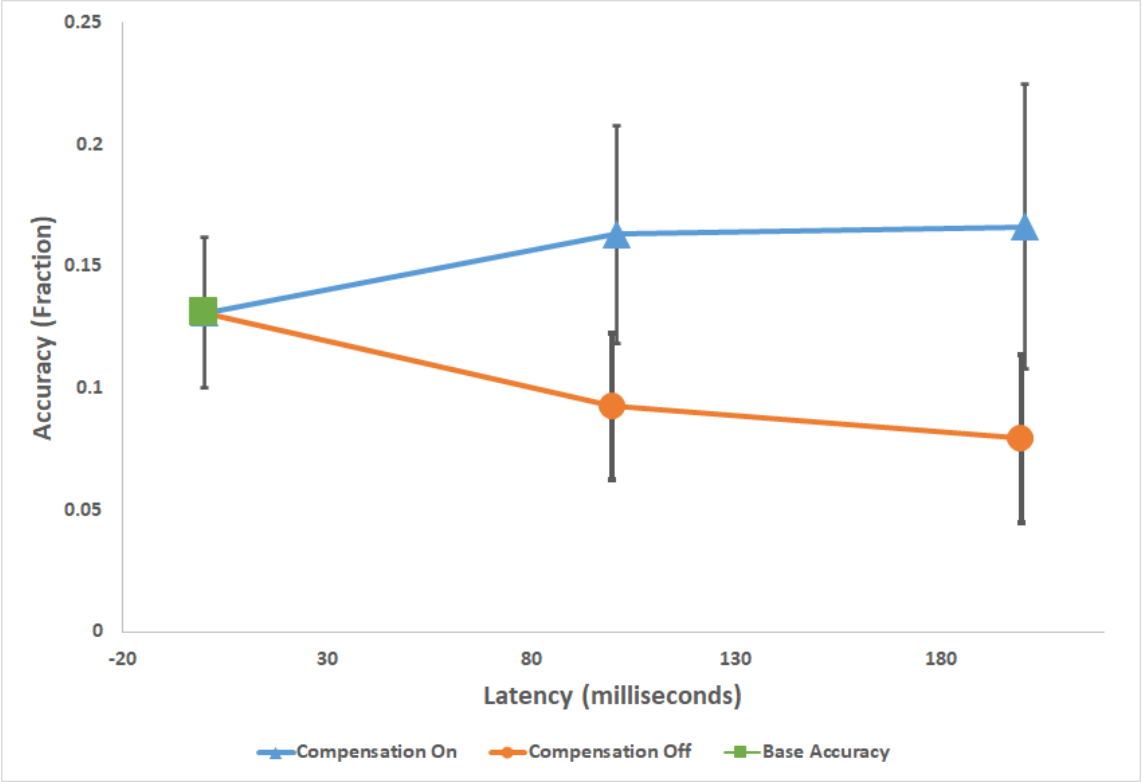


Figure 5.2.6: Average Fireball Accuracy for Test Group vs. Latency with Average Base Latency for 0ms

According to the figure, we can see that the "true" base accuracy for Fireball at 0 added latency is around 0.13. As latency increased, Fireball accuracy among the test group decreased from this average when hitbox scaling was off, while the accuracy increased with hitbox scaling on.

5.2.3 Accuracy vs. Latency: Lightning Strike

Similar to our analysis of the Fireball spell, we began with 126 data points from the 18 participants in the test group. After removing the data points from the tutorial matches, we had 108 data points. We then removed points in which 0 Lightning Strikes were cast, which reduced the number of data points to 50. Taking these remaining data points with 0 accuracy, we calculated a cut-off point in a manner similar to how we calculated the cut-off point for Fireball.

We removed data points where Lightning Strike was cast less than or equal to 2 times that had an accuracy of 0. This removed an additional 10 data points, leaving 40 data points remaining. Figure 5.2.7 contains the cumulative distribution function before and after this data removal took place.

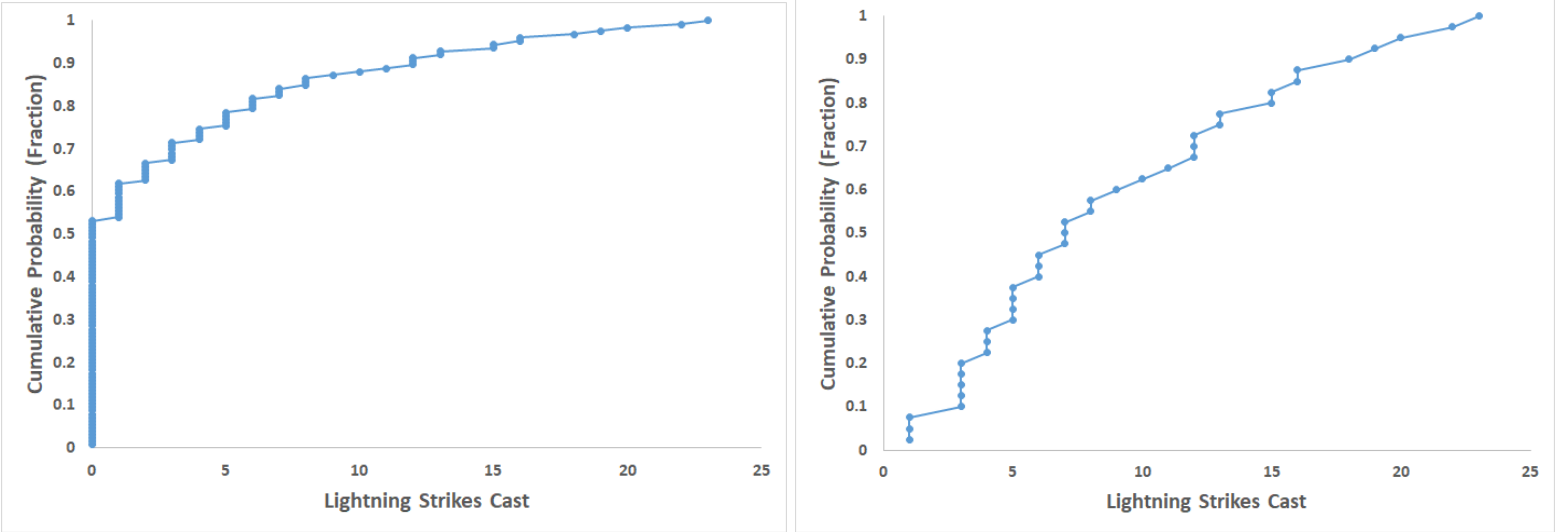


Figure 5.2.7: Cumulative Distribution of Lightning Strikes Cast Across All Trials from Test Group (left) [n=126] and Across All Trials from Test Group After Data Removal (right) [n=40]

From these figures, we can see that the data removal was able to remove the large left tail that was present in the cumulative derivation function for all the data. The function after data removal indicates that the distribution of Lightning Strikes cast is slightly normal. With regards to the accuracy of the spell, Figure 5.2.8 details the cumulative distribution of the remaining 40 data points in the test group, while Figure 5.2.9 is the corresponding QQ plot.

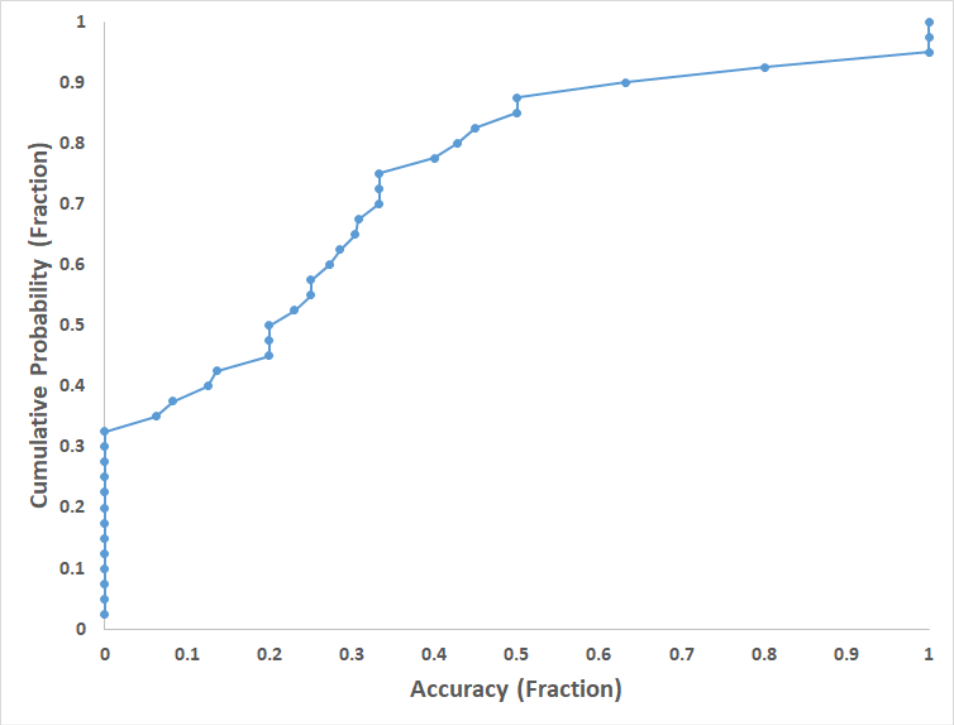


Figure 5.2.8: Cumulative Distribution of Lightning Strike Accuracies Across All Trials for the Test Group

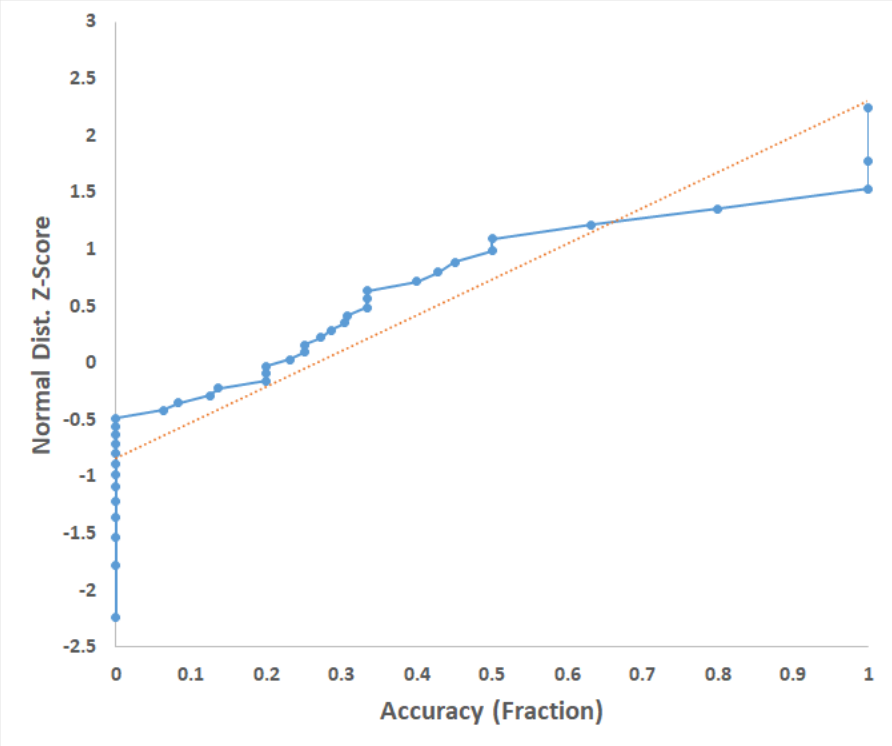


Figure 5.2.9: QQ Plot of Lightning Strike Accuracies Across All Trials for the Test Group

Other than the tails in each, both figures indicate a somewhat uniform distribution of accuracies. This matches the individual cumulative distributions and QQ plots of accuracies for Lightning Strike under the six individual trial conditions (see Appendices 5.C and 5.D).

We then graphed the average accuracy for Lightning Strike for each of the six trial conditions, along with 90% Confidence Intervals. This information can be seen in Figure 5.2.10.

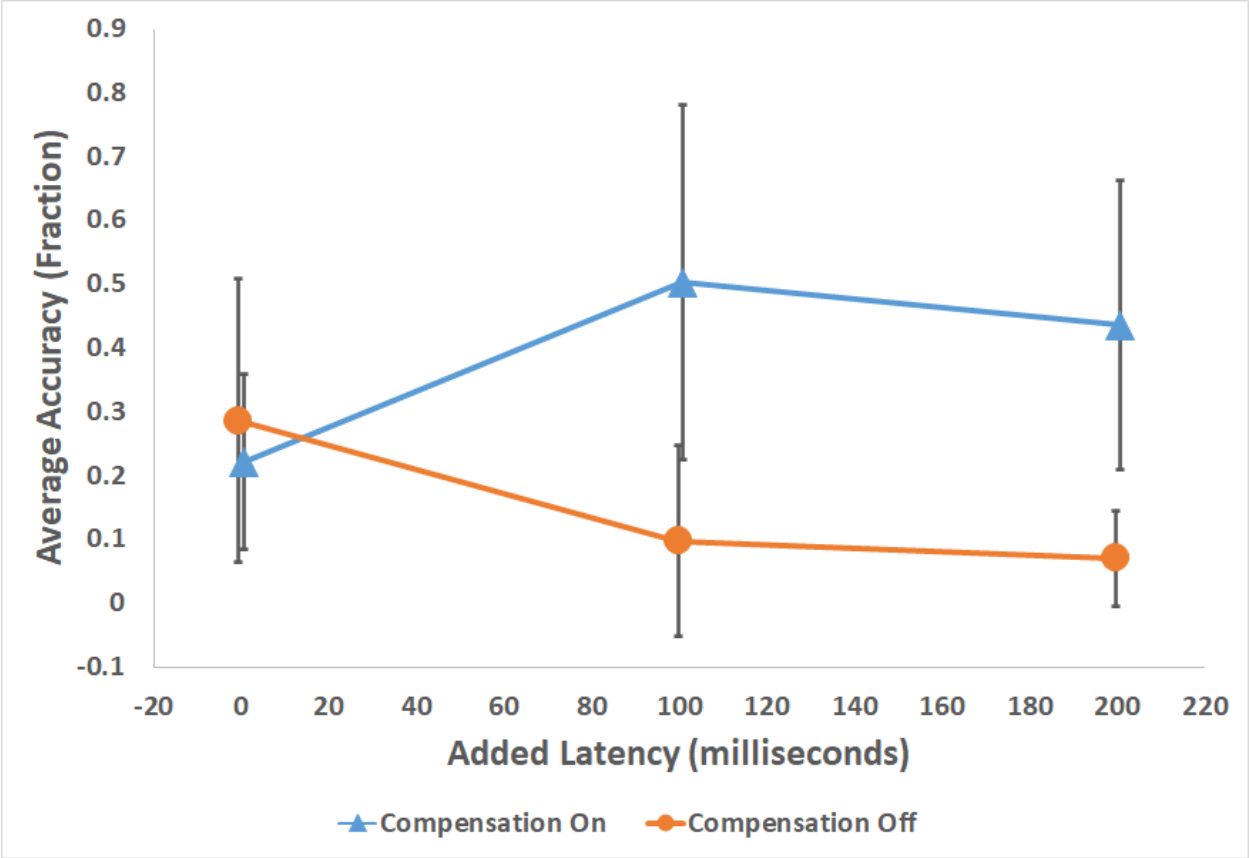


Figure 5.2.10: Lightning Strike Accuracy vs. Latency for Test Group

According to the figure, we can see similar accuracies at 0ms between compensation on and off. For 100ms and 200ms, the average accuracy with compensation on is greater than that with compensation off. To determine if these differences are statistically significant, we ran a t-test with unequal variance, similar to what was done with Fireball. The results of the t-tests can be seen in Table 5.2.3. The p-values are one-tailed p-values.

Added Latency	Degrees of Freedom	T-Value	P-Value
0	11	-0.47	0.32
100	9	2.52	0.02
200	5	3.22	0.01

Table 5.2.3 T-Test Between Lightning Strike Accuracies With Compensation On and Off, Alpha = 0.10

Based on the results in Table 5.2.3, we can see that there is no statistically significant difference between the average accuracies for 0ms with compensation on and off. However, for 100 and 200ms, the average accuracy with compensation on was statistically higher than that with compensation off.

In analyzing the Lightning Strike accuracy of the non-lagged group, we removed 32 out of 48 data points for having 0 Lightning Strikes cast. When calculating a cut-off point similar to how we did for Fireball, the cutoff value came out to be 0, so no more data points were removed. With the remaining 16 data points, the average Lightning Strike accuracy was 0.04 with 90% Confidence Interval of [-0.02, 0.10]. Similar to the average Fireball accuracy for the non-lagged group, we ran t-tests between the non-lagged group's average Lightning Strike accuracy and each of the test group's average Lightning Strike accuracies with an alpha value of 0.05. The results can be found below in Table 5.2.4. The p-values are two-tailed p-values.

Added Latency	Compensation On?	Degrees of Freedom	T-Value	P-Value
0	Yes	9	-2.34	0.04
0	No	8	-2.02	0.08
100	Yes	7	-3.16	0.02
100	No	7	-0.71	0.50
200	Yes	5	-3.56	0.02
200	No	14	-0.61	0.55

Table 5.2.4: T-Test Between Non-Lagged Group Average Lighting Strike Accuracy and Each of the Test Group's Average Fireball Accuracy, Alpha = 0.10

Based on this table, we can see that the average Lightning Strike accuracy was statistically different from the test group's accuracies for all trial parameters except for 100ms with compensation off and 200ms with compensation off. We can see in Figure 5.2.8 that those two points fall within the confidence interval for the non-lagged group's Lightning Strike accuracy confidence interval.

Similar to Fireball, we averaged together the Lightning Strike accuracies across test and non-lagged groups for 0ms, resulting in a total of 31 data points for 0ms. Figure 5.2.11 displays this average alongside the average Lightning Strike accuracies of the test group at 100 and 200ms with hitbox scaling on and off.

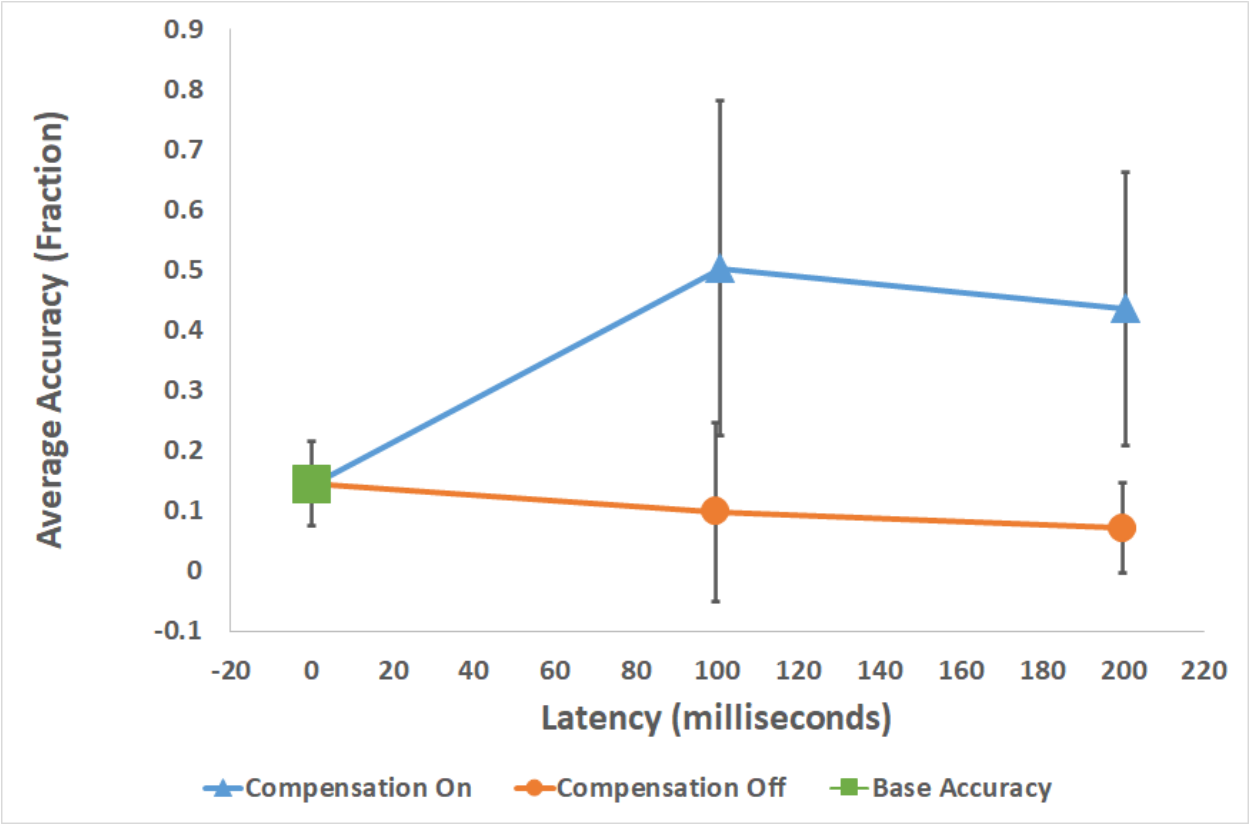


Figure 5.2.11: Average Lightning Strike Accuracy for Test Group vs. Latency with Average Base Latency for 0ms

From the figure, we can see that the average base Lightning Strike accuracy with 0ms of added latency is about 0.15. As latency increases, the average accuracy increases when hitbox scaling is on, but decreases slightly when hitbox scaling is off.

5.3 Quality of Experience

During the user study, participants were asked to complete a short post-trial survey after each 3-minute trial with the goal of measuring quality of experience under that trial's conditions.

The first question in the survey was borrowed from Stadia's own post-game questionnaire, which asks players to rate their experience based on the following five categories: Excellent, Good, Fair, Poor, or Bad. All participants were asked to rate their experience for all six trials (0-off, 0-on, 100-off, 100-on, 200-off, 200-on) using these categories. In order to analyze if these rating were affected by compensation for the participants that were not part of the non-lagged group, we assigned each of the categories a numeral equivalent: Excellent = 5, Good = 4, Fair = 3, Poor = 2, Bad = 1, and used these values to numerically represent the average experience of a player in each of the six trials. In Figure 5.3.1, we compared the average QoE (Quality of Experience) with compensation on for each latency value against the average QoE with compensation off. At 0ms of added latency we observed very little difference in QoE with latency compensation compared to without. At 100ms and 200ms of latency we observed decreases compared to 0ms with higher ratings with compensation on at each respective latency. As seen in the figure, there is a significant overlap for the averages for compensation on and compensation of, suggesting there is no difference in QoE.

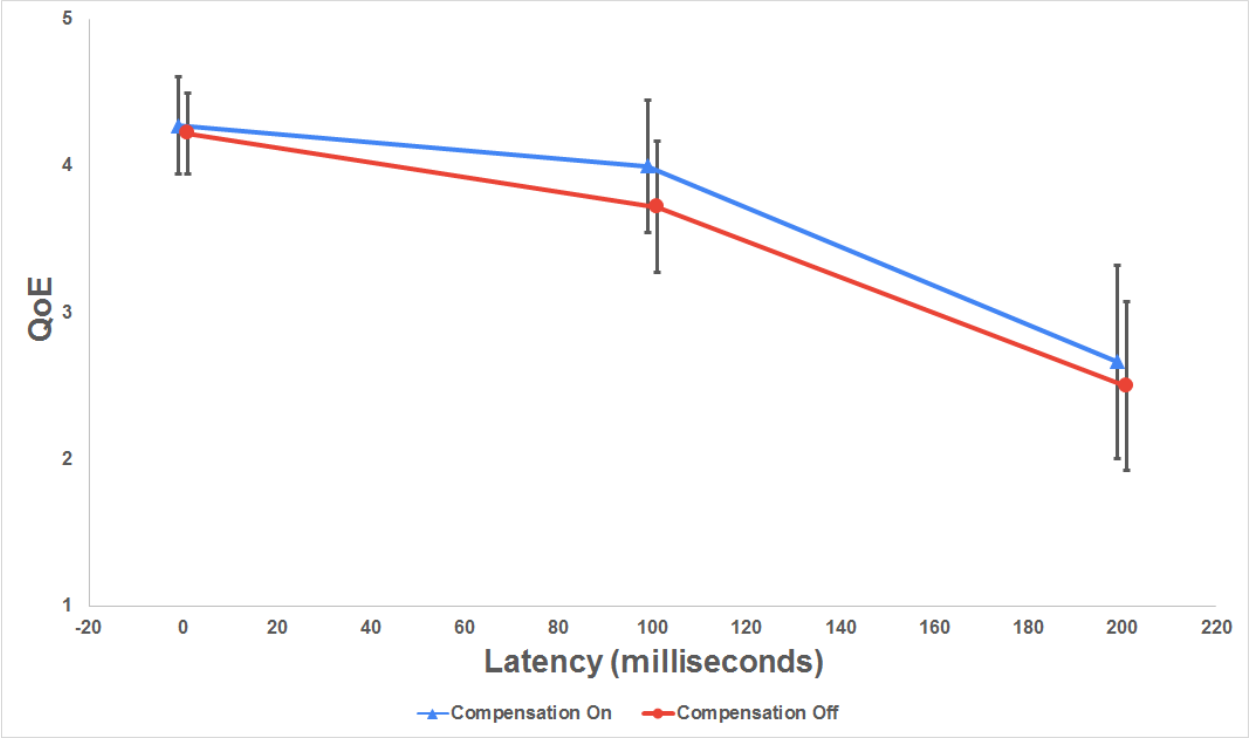


Figure 5.3.1: Average QoE (Quality of Experience) with Compensation on vs. Compensation off for each added latency category (0,100, and 200)

The difference in compensation on/off displayed in Figure 5.3.1 is insignificant when separated into the three latency categories, so we organized these average QoE values into strictly two groups: compensation on and compensation off. As shown in Figure 5.3.2, the difference of the total average of QoE when compensation was on compared to when compensation was off, was statistically insignificant as well.

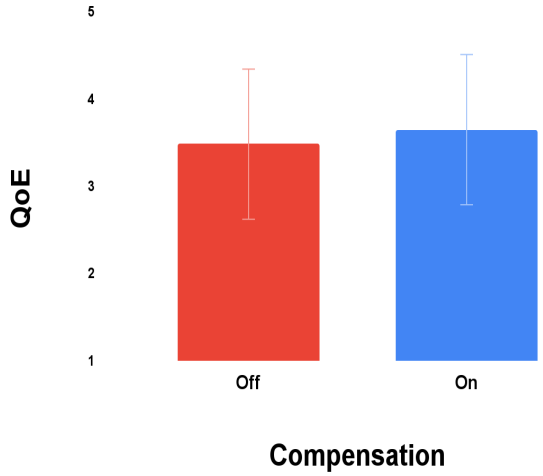


Figure 5.3.2 Average QoE with Compensation On vs. Off

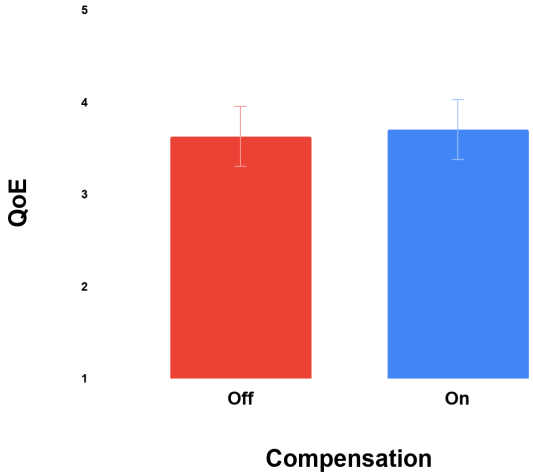


Figure 5.3.3 Average Perception of How Accurately the Game Reflected User Skill Across Trials

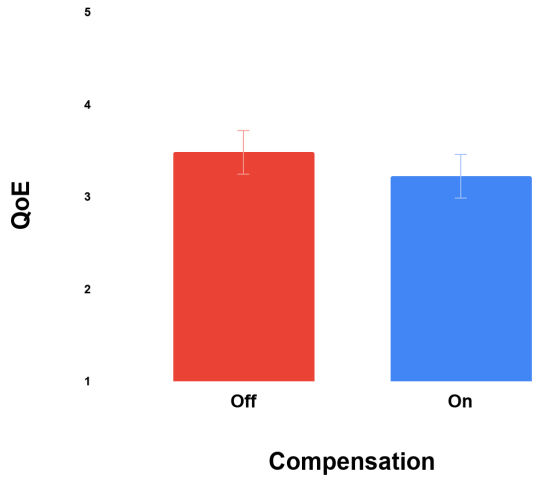


Figure 5.3.4 Average Perception of Fairness Across All Trials with Compensation on v.s. Compensation off for Test Group

Based on these analyses, we cannot conclude that our compensation technique had a significant impact on player QoE. However, we can see a slightly increased average QoE in both Figure 5.3.1 and Figure 5.3.2 in trials with latency compensation on. With a larger sample size the true difference in these values may show a difference.

We also asked some additional QoE questions in these post-trial surveys. These questions inquired about specific aspects of the users' experience. These questions included the following ratings: Responsiveness of Controls, Smoothness of Graphics, Game Accurately Reflects User Skill, Fairness of the Game, Would the User Play the Game Again Under These Conditions. The most significant of these ratings in aggregate was the user perception of how accurately the game reflected skill which displayed the largest difference between treatments of compensation on and compensation off as shown in Figure 5.3.3. Additionally, we observed a negative relationship between our latency compensation technique and the user perception of fairness on average as shown in Figure 5.3.4. Additional data on survey responses can be found in Appendix 5.E.

5.4: Gameplay Survey

After completing all six trials, all participants were asked to complete a Final Survey, which focused on feedback for our game.

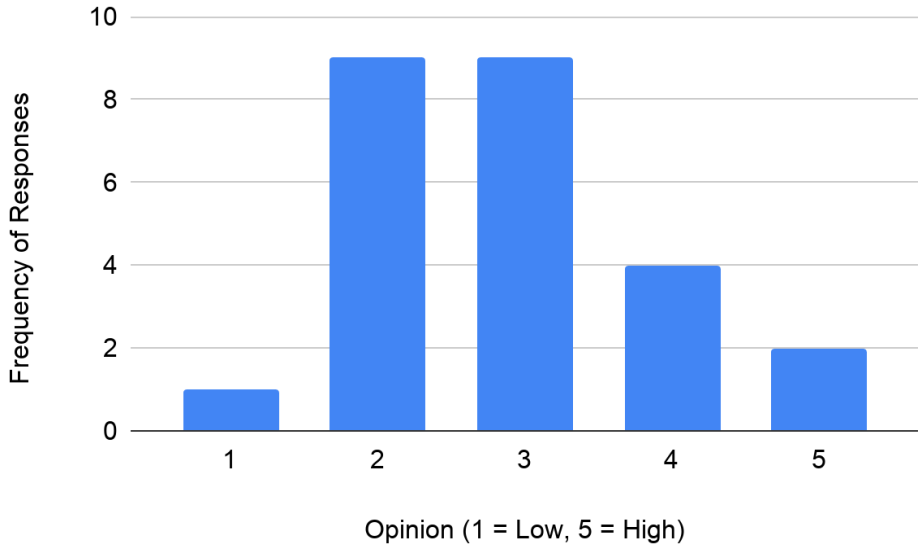


Figure 5.4.1: How Well Do You Believe the Bots Successfully Imitated an Average Human Player? (n=25).

As can be observed from Figure 5.4.1 and Figure 5.4.5, the average participant had mixed feelings on the human-ness of AI bots with the overall mean being 2.88. At the end of the survey, we included an additional feedback section that allowed participants freeform comment on the game. In the comments there were dissenting opinions on the AI bots: while it generally seems that the AI was somewhat competent, it was distinguishable from human players. For example, one participant described how the behavior of the AI stopping in order to fire made them seem unrealistic. Another participant stated that they would have had more fun if there were more non-AI players on the field.

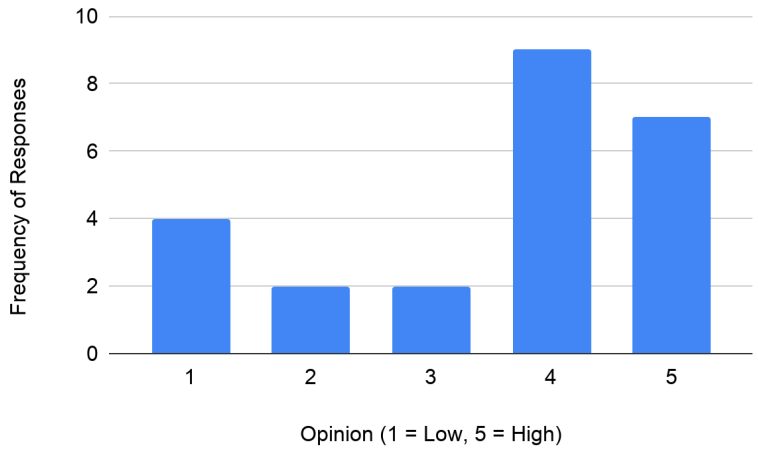


Figure 5.4.2. How Much did the Map Improve the Gameplay Experience? (n=24)

As can be observed from Figure 5.4.2 and Figure 5.4.5, participants generally thought that the map design improved the gameplay experience with a mean score of 3.54 with most participants picking a score of 4 or more. There were a few dissenting opinions by some participants as indicated by Figure 5.4.2 with one commenting that the map design was “barren”, but did not have any more substantive feedback.

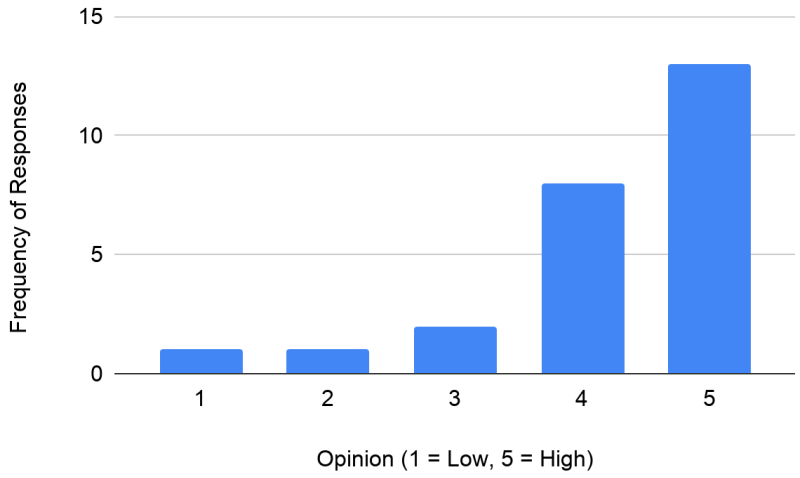


Figure 5.4.3: How Clear was the Objective of the Game? (n=25)

Seen by Figure 5.4.3, the average participant generally felt that the game conveyed its objective well with a mean score of 4.24 as seen by Figure 5.4.3.

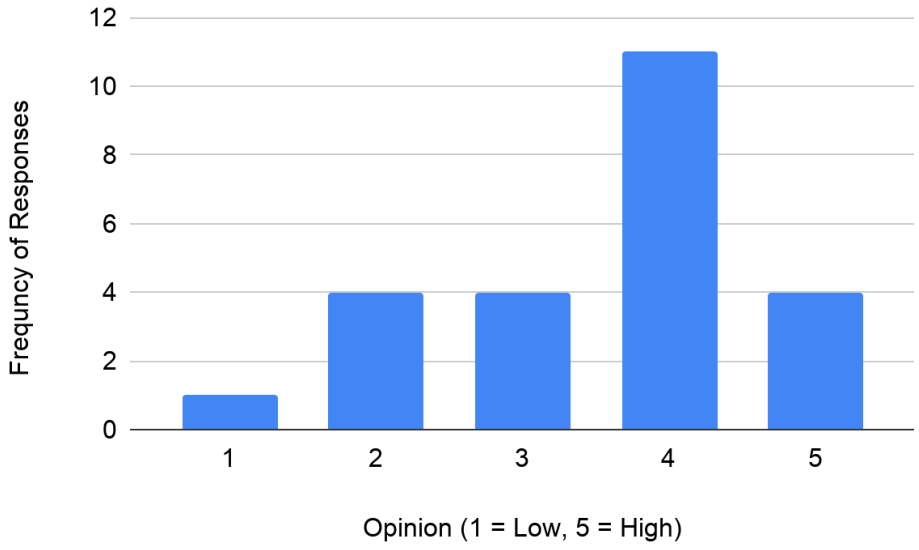


Figure 5.4.4: How Did You Feel about the Number of Spells? (n=24)

We gained a considerable amount of feedback from participants of the number of spells that are presently in the game. As seen by Figure 5.4.4, participants felt the number of spells was somewhat high with the mean response being 3.54. Additionally, some participants made comments indicating this because there was either too little time in the study to get used to all of them or that they kept forgetting spell combinations. This suggests that our game has a high learning curve, which implies participants might have required more time to get proficient with all spells, affecting our results.

Summary of Responses	Mean	Median	Standard Deviation
Q1: How well do you believe the bots successfully imitated an average human player?	3.38	3.00	2.13
Q2: How much did the map design improve the gameplay experience?	4	4	1
Q3: How clear was the objective of the game?	3.90	5.00	1.48
Q4: How do you feel about the number of spells?	3.50	4	1.15

Table 5.4.1. Summary of Statistics

5.5 Discussion

In this section, we interpret the results of the user study in order to draw conclusions about the success of hitbox scaling in Catalyst. We first interpret the results with regards to player performance. Afterwards, we discuss the subjective results from the QoE surveys and to discuss hitbox scaling's impact on player enjoyment of the experience.

5.5.1 Player Performance

To analyze player performance, we compared average accuracy for the Fireball and Lightning Strike spells under each of the different trial conditions. Figures 5.2.5 and 5.2.8 detail this comparison, while Tables 5.2.1 and 5.2.2 detail the results of the t-tests determining the statistical significance of the results. For both spells, we found a statistically significant improvement in average player accuracy at 100ms and 200ms of latency with hotbox scaling on compared to average accuracy without hitbox scaling. Additionally, neither spell had a statistically significant difference in average player accuracy at 0ms with or without hitbox scaling.

In comparing the average accuracies between test and non-lagged groups, we can see that the average Fireball accuracy of the test group was statistically similar to each of the non-lagged group accuracies for all trial parameters. This makes it difficult to draw any conclusions between the two groups with regards to Fireball accuracy. However, the test group's average Lightning Strike accuracy was statistically less than the non-lagged group's Lightning Strike accuracy for all conditions except for increased latency with no compensation. This suggests that hitbox scaling may help increase player performance past the average amount under no latency.

These results suggest that our latency compensation technique was successful at compensating for latency from an objective standpoint in that it was able to prevent a decrease in player performance in spite of increasing latency. However, Sabet et al. point out that the goal of a latency compensation technique should not be to alter how challenging a game is, but rather to keep it at the difficulty it would be if there was no latency [19]. In this way, the fact that player performance improved with hitbox scaling on as latency increased means that they may have an unfair advantage compared to other players and the game may be too easy for them. The scaling equation may need to be modified to scale hitbox more slowly so as to not prevent this from happening.

5.5.2 Quality of Experience

To measure our participants' quality of experience we asked them to complete a short post-trial survey after each 3-minute trial. The main metric of Quality of Experience (QoE) was a single question asked at the beginning of the survey: "How was your game session?". By aggregating and averaging the responses to this question we were able to analyze the participants' general QoE across all trials in general and with each condition. However, as can be

seen in Figure 5.3.2, there was no statistically significant difference in the average values of QoE with compensation compared to without compensation. A similar observation can be made when considering average QoE across specific trial conditions as in Figure 5.3.1.

This suggests that despite the improvements in performance noted, we cannot conclude that our latency compensation effect maintains Quality of Experience as latency increases. However, as noted before, the small improvement in QoE shown by our results, while not statistically significant, suggests that if the technique were tested with a larger sample size QoE differences may be shown to be significant.

Additionally, our post-trial survey included a number of questions related to specific aspects of the game. As shown in Figure 5.3.3, there was an observable (albeit statistically insignificant considering the error) increase in the user perception that the game accurately reflected their skill with latency compensation on. We also observed, in Figure 5.3.4, a decrease (albeit statistically insignificant) in user perception of fairness with latency compensation on.

These results suggest that it is possible that the latency compensation technique improves the user's perception that their own skill is reflected accurately while also decreasing their perception that the game is fair considering that the circumstances are skewed either for or against them. This is a potential limitation of the latency compensation technique as alluded to by the player performance data and the discussion thereof.

5.5.3 Gameplay Survey

The results also indicate aspects that we can improve our game and future studies. First, our results suggest that our AI needs more work in order for them to accurately emulate human players. Passive observations and player feedback suggested this mainly was due to the unrealistic paths that they would travel throughout the map as well as stopping in order to fire at the player. One approach that could be done would be to make the AI more realistic to emulate the average player, which might not be feasible in a short time. For a future study, hopefully once current circumstances allow participants to play in a six-player match, we might be able to create a more realistic gameplay experience with only human players. Another advantage to this approach would be that we could test the latency compensation technique across more players and through more complex gameplay interactions.

Another result that can be ascertained is that the map created was possibly sufficient for the study and for an enjoyable gameplay experience. However, it appears that the map design can be improved because we received some indication that the map was lacking in obstacles. This constructive criticism can be addressed by potentially including more obstacles either in the middle of the map and the sides to make matches more interesting by increasing tension in the players.

Through our data, it is possible that the objective of the game was not clear. Moreover, this result may be due to confounding factors such as the lack of an in-game tutorial as well as the facilitators of the study stating the objective of the game beforehand. This sentiment was expressed by a participant as seen in Appendix 5.F who recommended the inclusion of a tutorial.

Finally, we received feedback on the number of spells, which could indicate that the gameplay experience may be improved by reducing the number of spells. This could also be alleviated by including a tutorial demonstrating how each spell works and how to invoke each.

6. Conclusions

Latency has been shown to negatively affect a player's experience when gaming [17]. This latency can come from several sources including the device itself [5], but network latency is often higher and less consistent compared to other sources of latency [14]. Network latency will be experienced by all users with cloud gaming because of the platform's architecture [4]. With the increasing accessibility and prevalence of cloud gaming in the gaming space, many companies and developers are considering the issue of network latency and its effects of cloud gaming [1, 4].

While increasing the network speeds of all users is outside of the control of cloud gaming platforms, latency compensation techniques can mitigate the negative effects of latency. Attribute scaling latency compensation techniques focus on maintaining or improving user performance in environments with significant latency [7, 9]. However, the attribute scaling techniques have not been thoroughly tested in a cloud gaming environment [7, 9, 11]. Given this gap in the research, evaluating the efficacy of an attribute scaling technique can provide insight into the future of latency compensation for cloud gaming.

We created a game called Catalyst, a multiplayer first-person shooter game with spell-based combat, to assess attribute scaling by scaling the hitboxes of players. We then conducted a user study over the course of two weeks, collecting data related to player accuracy as well as the opinions of the participants on topics relating to Quality of Experience (QoE).

To analyze player performance, we compared average accuracy for the Fireball and Lightning Strike spells under each of the different trial conditions and then compared those measurements to the average accuracy of the control group, who was not subjected to added latency. Overall, we observed a 76.4% and 109.8% improvement in player accuracy for Fireball

when latency increased to 100ms and 200ms respectively with compensation on. The improvement for Lightning Strike was 414.8% and 518% at 100ms and 200ms respectively with compensation. At 0ms of latency, there was no statistically significant difference in accuracies when compensation was on or off. The statistically significant change in accuracy when hitbox scaling is used suggests that the technique may maintain player performance as latency increases.

After each of the six trials, we asked participants to complete a short post-trial survey, which focused on player QoE. The first question was the same as Stadia's post-game survey, which asks users to summarize their experience into one out of five categories: Excellent, Good, Fair, Poor, or Bad. To analyze if hitbox scaling had a significant impact on player QoE, we assigned numerical equivalents to these categories (5-Excellent, 4-Good, 3-Fair, 2-Poor, 1-Bad) and compared the average QoE for each latency category (0 ms added latency, 100 ms added latency, and 200 ms added latency) with compensation on versus compensation off. The difference between these two scenarios did not have a statistically significant difference, implying hitbox scaling may not have an effect on player QoE. However, although not statistically significant, there was a slight improvement in QoE when compensation was on, suggesting that with a larger sample size, significance might emerge.

In addition to the question about general QoE our post-trial survey included several questions about specific aspects of the user's experience during that trial. Each of these questions was on a five-point scale similar to the general question. We were then able to aggregate these responses and average them to compare the users' perception during trials with compensation on compared to those with compensation off. As for the general question, we observed no statistically significant difference between these conditions although we were able to note that

user perception of skill was higher with compensation on and user perception of fairness was lower with compensation on.

Based on our results, with hitbox scaling we did not see a significant improvement in QoE, but we did see a significant increase in player accuracy. Further research is required into the efficacy of hitbox scaling as an effective latency compensation technique. Our research was limited due to COVID-19 restrictions, which naturally led to lower participation in our user studies, leading to a smaller sample size. While AI bots were somewhat successful in acting as teammates, they can not contribute in the same way as a human teammate and similarly cannot provide useful performance and QoE data. In the case of Catalyst, having six participants play a full match could provide data regarding how hitbox scaling performs under complex multiplayer interactions.

7. Future Work

In this section we identify multiple ways to extend and continue the research presented in this paper. These methods include short, medium, and long term research ideas and propositions.

Short Term Work

The first way to continue research is to evaluate full matches of Catalyst with three human players on each team. Due to COVID restrictions we were only able to have one player on each team with two AI bots, since participants needed to play the game in-person to maintain equal computer and network conditions. However, having all human players in a game would better represent a multiplayer situation with a cloud gaming service, therefore producing more relevant results.

Another method also involving Catalyst would be to use different alpha values in the exponentially weighted moving average used to provide a measure of latency and then scale player hitboxes. Using higher alpha values weighs the previous data more heavily, smoothing out the average and making it less reactive to spikes in latency. We used an alpha of 0.99 in our study, meaning the average was fairly steady, although there was not much variation in latency values in most trials. Further research could be done to determine how lower alpha value results change in accuracy for hitbox scaling compensation, and could be combined with more varied network conditions.

One final idea for short term work is to modify the formula used to calculate the scale value. We based this on the equation given to us by the IQP study, with modifications done through testing to increase the scaling [22]. Further adjustments could be made based on what we

gathered in our evaluation study. Changing this equation would certainly affect player accuracy since this is the basis of the hitbox scaling compensation technique.

Medium Term Work

Another area of future work we did not have time to develop was an API built into Unreal Engine 4 that would allow game developers to more easily incorporate hitbox scaling into their games. This would enable more widespread applications of the technique, also providing more data to determine whether it is effective in offsetting high latency conditions.

Another way to extend or study using Catalyst could be to modify the game to run a more controlled study to create a new compensation equation that accounts for the distance between the player and the target they are aiming at. The equation used in this study was based on a formula that did not take this factor into account, since in the IQP study the player was at a relatively fixed distance from the targets throughout trials [24]. However, in the full game of Catalyst, the map is large and players can be much farther away from each other, or they can fight very close together as well. An equation taking these factors into account may yield better results in the full game such as speed and distance to determine difficulty.

Long Term Work

A final idea to further this research in the long term would be to evaluate other latency compensation techniques in cloud gaming platforms. This could include other forms of attribute scaling, as well as completely different varieties such as aim assistance or time related techniques. More studies could be done with both Catalyst or other games to determine which of these approaches can be effective in compensating for latency in cloud gaming.

8. References

1. Chun-Ying Huang, Cheng-Hsin Hsu, Yu-Chun Chang, and Kuan-Ta Chen. 2013. GamingAnywhere: an open cloud gaming system. In *Proceedings of the 4th ACM Multimedia Systems Conference (MMSys '13)*. Association for Computing Machinery, New York, NY, USA, 36–47. DOI: https://doi.org/10.1145/2483977.2483981

2. Global cloud gaming market size 2021. gamesindustry.biz. Retrieved April 30, 2021 from https://www.statista.com/statistics/932758/cloud-gaming-market-world/

3. Google. Stadia - One place for all the ways we play (google.com). Retrieved 30 April, 2021 from https://stadia.google.com.

4. Sharon Choy, Bernard Wong, Gwendal Simon, and Catherine Rosenberg. 2012. The brewing storm in cloud gaming: a measurement study on cloud to end-user latency. In *Proceedings of the 11th Annual Workshop on Network and Systems Support for Games (NetGames '12)*. IEEE Press, Article 2, 1–6. DOI: https://doi.org/10.1109/NetGames.2012.6404024

5. De-Yu Chen, Hao-Tsung Yang, and Kuan-Ta Chen. 2013. Dude, the source of lags is on your computer. In *Proceedings of Annual Workshop on Network and Systems Support for Games (NetGames '13)*. IEEE, Denver, CO, USA, 1–6. DOI: 10.1109/NetGames.2013.6820608

6. Michael Jarschel, Daniel Schlosser, Sven Scheuring, and Tobias Hoßfeld. 2013. Gaming in the clouds: QoE and the users' perspective. *Mathematical and Computer Modelling* 57, 11–12 (June 2013), 2883-2894, DOI: https://doi.org/10.1016/j.mcm.2011.12.014

7. Saeed Shafiee Sabet, Steven Schmidt, Saman Zadtootaghaj, Carsten Griwodz, and Sebastian Moller. 2018. Towards applying game adaptation to decrease the impact of delay on quality of experience." In Proceedings of the IEEE International Symposium on Multimedia (ISM). IEEE, Taichung, Taiwan, 114-121. DOI: https://doi.org/10.1109/ISM.2018.00028

8. Mark Claypool and Kajal Claypool. 2006. Latency and player actions in online games. *Commun. ACM* 49, 11 (November 2006), 40–45. DOI: https://doi.org/10.1145/1167838.1167860

9. Zenja Ivkovic, Ian Stavness, Carl Gutwin, and Steven Sutcliffe. 2015. Quantifying and mitigating the negative effects of local latencies on aiming in 3D shooter games. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. Association for Computing Machinery, New York, NY, USA, 135–144. DOI: https://doi.org/10.1145/2702123.2702432

10. Cheryl Savery and T. C. Nicholas Graham. 2013. Timelines: Simplifying the programming of lag compensation for the next generation of networked games. *Multimedia Systems* 19, 3 (Jun. 2013), 271-287. DOI: https://doi.org/10.1007/s00530-012-0271-3

11. Michael Long and Carl Gutwin. 2018. Characterizing and modeling the effects of local latency on game performance and experience. In *Proceedings of the 2018 Annual Symposium on Computer-Human Interaction in Play (CHI PLAY '18)*. Association for Computing Machinery, New York, NY, USA, 285–297. DOI: https://doi.org/10.1145/3242671.3242678

12. Scott Bateman, Regan Mandryk, Carl Gutwin, and Robert Xiao. 2009. Investigation of targeting-assistance techniques for distant pointing with relative ray casting. Technical Report #2009-03. Department of Computer Science, University of Saskatchewan, Saskatoon, Canada.

13. Michael Long and Carl Gutwin. 2019. Effects of local latency on game pointing devices and game pointing tasks. In *Proceedings of the CHI Conference on Human Factors in Computing Systems* (CHI '19), Association for Computing Machinery, New York, NY, USA, 1–12. DOI: https://doi.org/10.1145/3290605.3300438

14. James Anouna, Zachary Estep, Michael French. 2014. Network latency and cloud games: A study using GamingAnywhere. Interactive Qualifying Project. Worcester Polytechnic Institute.

15. Mark Claypool, Tianhe Wang, and McIntyre Watts. 2015. A taxonomy for player actions with latency in network games. In *Proceedings of the 25th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '15)*. Association for Computing Machinery, New York, NY, USA, 67–72. DOI: https://doi.org/10.1145/2736084.2736093

16. Saeed Sabet, Steven Schmidt, Carsten Griwodz, & Sebastian Möller. 2019. Towards the impact of gamers' adaptation to delay variation on gaming quality of experience. In *Proceedings of the Eleventh International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE, Berlin, Germany, 1-6. DOI: 10.1109/QoMEX.2019.8743239.

17. Mark Claypool. 2018. Game input with delay—Moving target selection with a game controller thumbstick. ACM Trans. Multimedia Comput. Commun. Appl. 14, 3s, Article 57 (August 2018), 22 pages. DOI: https://doi.org/10.1145/3187288

18. Injung Lee, Sunjun Kim, and Byungjoo Lee. 2019. Geometrically compensating effect of end-to-end latency in moving-target selection games. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19)*. Association for Computing Machinery, New York, NY, USA, Paper 560, 1–12. DOI: https://doi.org/10.1145/3290605.3300790

19. Saeed Shafiee Sabet, Steven Schmidt, Saman Zadtootaghaj, Carsten Griwodz, and Sebastian Moller. 2018. Towards applying game adaptation to decrease the impact of delay on quality of experience, in *Proceedings of the IEEE International Symposium on Multimedia (ISM)*. IEEE, Taichung, Taiwan, 114-121. DOI: https://doi.org/10.1109/ISM.2018.00028

20. Ryan Shea, Jiangchuan Liu, Edith C.-H. Ngai, and Yong Cui. 2013. Cloud gaming: architecture and performance. *IEEE Network*, 27, 4 (July-Aug 2013), 16–21. DOI: 10.1109/mnet.2013.6574660.

21. Mark Claypool and David Finkel. 2014. The effects of latency on player performance in cloud-based games. In *Proceedings of the 13th Annual Workshop on Network and Systems Support for Games (NetGames '14)*. IEEE Press, Article 2, 1–6.

22. Saeed Shafiee Sabet, Steven Schmidt, Saman Zadtootaghaj, Carsten Griwodz, and Sebastian Möller. 2020. Delay sensitivity classification of cloud gaming content. In *Proceedings of the 12th ACM International Workshop on Immersive Mixed and Virtual Environment Systems (MMVE '20)*. Association for Computing Machinery, New York, NY, USA, 25–30. DOI: https://doi.org/10.1145/3386293.3397116

23. Marc Carrascosa, and Boris Bellalta. 2020. Cloud-gaming: Analysis of Google Stadia traffic. Retrieved from https://arxiv.org/abs/2009.09786

24. Edward Carlson, Tian Fan, and Zijian Guan. 2021. Towards Usable Attribute Scaling for Latency Compensation in Cloud-based Games. Interactive Qualifying Project. Worcester Polytechnic Institute.

25. Proletariat, Inc. 2020. Spellbreak. Game [PC]. (3 September 2020). Proletariat, Inc, Boston, Massachusetts, USA.

26. Valve Corporation. 2013. DOTA 2. Game [PC]. (9 July 2013). Valve Corporation, Bellevue, Washington, USA.

27. Rick Brewster. 2004. Paint.net, dotPDN, LLC, https://www.getpaint.net/

28. Vecteezy. 2007. Free Vector Art https://www.vecteezy.com/

29. Tokenstamp 2014. http://rolladvantage.com/tokenstamp/

30. Epic Games. 2010. Infinity Blade. Game [iOS]. (9 December 2010). Epic Games, Potomac, Maryland, USA.

31. Epic Games. 2016. Paragon. Game [PC]. Epic Games, Potomac, Maryland, USA.

32. MIT. 2014. Clumsy, https://github.com/jagt/clumsy

9. Appendices

Appendix 3.A: Asset List

Name	Type	Source
Spell Animations	Animation	Mixamo.com
Death Animation	Animation	Mixamo.com
Spell Effects	FX	FXVarietyPack - UE Marketplace
Butterfly Wings Model	Meshes	https://www.turbosquid.com/3d-models/3d-set-fairy-butterfly-wings-1297794
Concrete Walls	Meshes	Bunker - Starlight Arts
Crunch Character Model	Meshes	Paragon
Foliage and Vegetation	Meshes	Infinity Blade Grasslands Pack - UE Marketplace
Greystone Character Model	Meshes	Paragon
Ground Assets	Meshes	Unreal Engine 4 Starter Content
Iggy and Scorch Character Model	Meshes	Paragon
Infinity Blade Character Meshes	Meshes	Infinity Blade Warriors Pack - UE Marketplace
Kallari Character Model	Meshes	Paragon
Khaimera Character Model	Meshes	Paragon
Lamps	Meshes	Bunker - Starlight Arts
Metal Platforms	Meshes	Bunker - Starlight Arts
Muriel Character Model	Meshes	Paragon
Phase Character Model	Meshes	Paragon
Ruins Assets	Meshes	Infinity Blade Grasslands Pack - UE Marketplace, Unreal Engine 4 Starter Content
The Fey Character Model	Meshes	Paragon
Torches	Meshes	Infinity Blade Grasslands Pack - UE Marketplace
Water	Meshes	Unreal Engine 4 Starter Content
Background Music - Offensive	Music	Ashamaluev Music - https://ashamaluevmusic.fanlink.to/dDDQ
Cannonball Splash A2 Sound Effect	Sound	https://www.fesliyanstudios.com/royalty-free-sound-effects-download/water-splashing-20

Chimes Sound (Wing Formation)	Sound	https://orangefreesounds.com/chimes-sound-effect/
Energy Beam	Sound	http://freesoundeffect.net/sound/sci-fi-energybeam-295-sound-effect
Fire/explosion sound effect	Sound	Unreal Engine 4 Starter Content
Flame Ignition	Sound	https://freesound.org/people/hykenfreak/sounds/331621/
Heavy Splash A2 Sound Effect	Sound	https://www.fesliyanstudios.com/royalty-free-sound-effects-download/water-splashing-20
Laser Static 05	Sound	http://freesoundeffect.net/sound/laser-static-05-sound-effectt
Laser Static FX	Sound	http://freesoundeffect.net/sound/laser-static-05-sound-effect
Magic Fairy Dust (Wing Release)	Sound	https://orangefreesounds.com/magic-fairy-dust-sound-effect/
Metal Hammer Anvil Hit A1	Sound	https://www.fesliyanstudios.com/royalty-free-sound-effects-download/hammer-hitting-metal-57
Sci-fi Energy Beam 295	Sound	http://freesoundeffect.net/sound/sci-fi-energybeam-295-sound-effect
Sci-fi Explosion 20	Sound	Ryan VanEerde - https://www.zapsplat.com/music/science-fiction-small-build-up-into-explosion-deep-reverb-futuristic/
Sword Scraping Shield	Sound	https://freesfx.net/sfx/Metal-Shield
Thunder Sound FX	Sound	http://soundbible.com/2053-Thunder-Sound-FX.html
Fairy Wings Icon	UI	https://depositphotos.com/37524523/stock-illustration-pair-of-pink-fairy-wings.html
Fire Border	UI	https://toppng.com/free-image/flame-dragon-fire-red-fire-dragon-circle-PNG-free-PNG-Images_164429
Fire Icon	UI	https://www.vecteezy.com/vector-art/623462-fire-logo-template-vectors
Fire Tornado Vector Icon	UI	https://www.shutterstock.com/image-vector/fire-tornado-symbol-vector-logo-sign-1486851239
Ice Cube Vector Icon	UI	https://www.vecteezy.com/vector-art/295927-piece-of-ice-cube-melting
Lightning Border	UI	https://www.freepik.com/free-vector/circle-lightning-realistic-transparent-background_5971443.htm
Lightning Icon	UI	https://www.vecteezy.com/vector-art/540389-lightning-bolt-icon
Magic Reticle	UI	https://toppng.com/magic-circle-PNG-free-PNG-Images_274075
Masquerade Mask Icon	UI	http://clipart-library.com/masquerade-mask-cliparts.html
Roman Shield Clip Art	UI	https://www.clipartkey.com/view/TTixJR_clip-art-romans-shields-roman-shield-transparent/
Water Border	UI	https://www.hiclipart.com/free-transparent-background-png-clipart-immii
Water Drop Icon	UI	https://www.vecteezy.com/vector-art/599373-water-drop-logo-template-vector

Appendix 4.A: Demographics Survey

1. What is your Player ID? (ask the proctor if you are not sure)
2. What is your age?
3. What gender do you identify with?
    a. Female
    b. Male
    c. Non-binary
    d. Prefer not to say
    e. Other: _____
4. What is your major(s)? Select all that apply.
    a. Computer Science
    b. Interactive Media and Game Development
    c. Data Science
    d. Other: _____
5. What is your minor(s)? Select all that apply
    a. Computer Science
    b. Interactive Media and Game Development
    c. No minor
    d. Other: _____
6. What is your graduation year?
7. What is your preferred gaming platform?
    a. PC
    b. Console
    c. Mobile
    d. Cloud Gaming
    e. Other: _____
8. Rate your proficiency using mouse and keyboard for games. (1-5; low to high)
9. What is the average hours per week you spend playing video games?
    a. I usually don't play video games
    b. Less than 1 hour
    c. 1-5 hours
    d. 6-10 hours
    e. 11-15 hours
    f. More than 15 hours
10. Rate your skill with FPS (First-Person Shooter) games. (1-5; low to high)
11. What is your experience with cloud gaming platforms?
    a. No cloud gaming experience
    b. Experience with cloud gaming
    c. I don't know
12. If you had experience with cloud gaming, rate your experience with it. (1-5; low to high)
13. If you had experience with cloud gaming, have you ever tried Google Stadia?
    a. Yes
    b. No

Appendix 4.B: Post-Trial Survey

1. What is your Player ID? (ask the proctor if you are not sure)
2. How was your game session?
    a. Excellent
    b. Good
    c. Fair
    d. Poor
    e. Bad
3. Rate the responsiveness of the controls. (1-5; low to high)
4. Rate the smoothness of the graphics. (1-5; low to high)
5. Rate how well you felt your performance in the game accurately reflected your skill. (1-5; low to high)
6. Rate how unfair the game felt. (1-5; low to high)
7. Rate your likelihood to play the game again under these conditions. (1-5; low to high)
8. Any additional comments?

Appendix 4.C: Final Survey

1. What is your Player ID? (ask the proctor if you are not sure)
2. How well do you believe the bots successfully imitated an average human player? (1-5; low to high)
3. How much did the map design improve the gameplay experience? (1-5; low to high)
4. How clear was the objective of the game? (1-5; low to high)
5. How do you feel about the number of spells? (1-5; low to high)
6. Are you completing this study for Psychology research credit, IMGD playtesting credit, or to participate in the gift card raffle? (select all that apply)
    a. No
    b. Psychology Research Credit
    c. IMGD Playtesting Credit
    d. Gift card raffle!
7. If yes, enter your full name and WPI email below.
8. Do you have any additional comments?

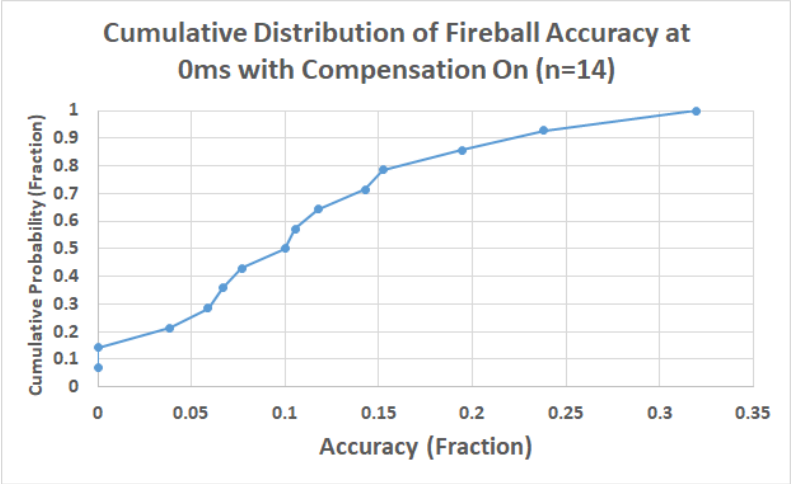
Appendix 4.D: Catalyst Recruitment Trailer

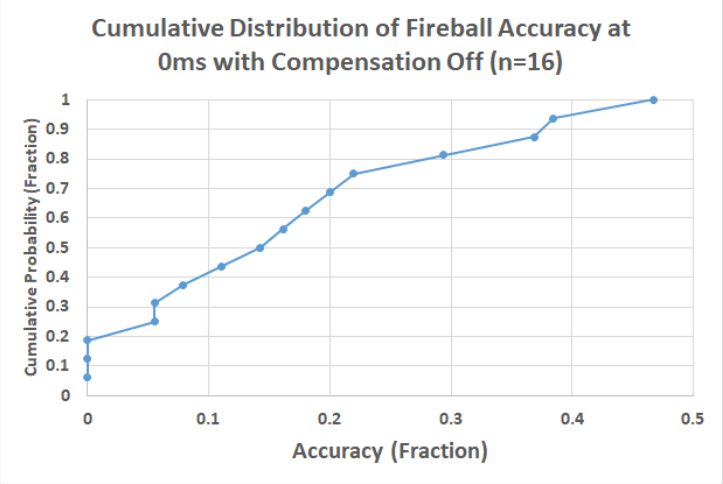
https://www.youtube.com/watch?v=1Gw2d4qWCUA

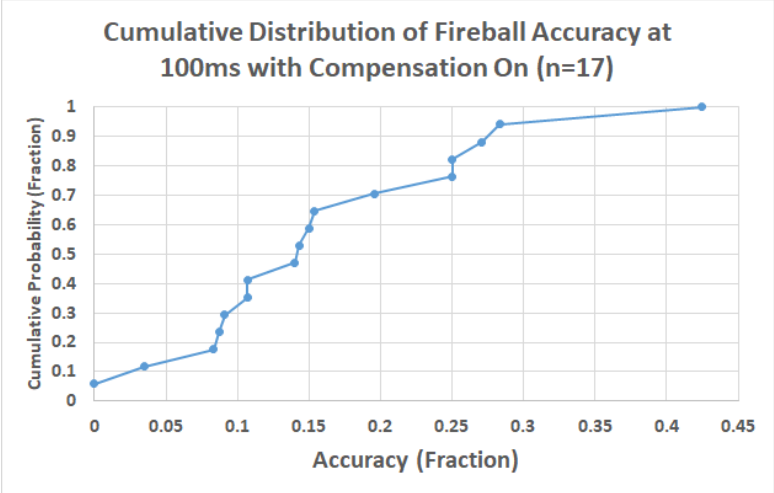
Appendix 4.E: Sample Log File Excerpt

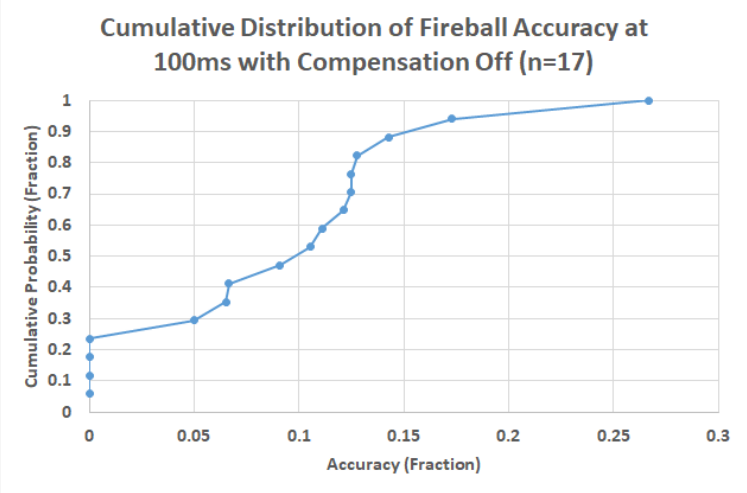
```
2021-04-16T20:29:00.761Z | LATENCY | TimeLeft : 165 | PlayerName : P041605 | Latency : 214.962006 ms | AverageLatency : 219.164062 ms
2021-04-16T20:29:00.783Z | LATENCY | TimeLeft : 165 | PlayerName : J041606 | Latency : 9.766000 ms | AverageLatency : 10.054724 ms
2021-04-16T20:29:01.769Z | LATENCY | TimeLeft : 164 | PlayerName : P041605 | Latency : 214.964005 ms | AverageLatency : 219.122070 ms
2021-04-16T20:29:01.792Z | LATENCY | TimeLeft : 164 | PlayerName : J041606 | Latency : 11.350000 ms | AverageLatency : 10.067677 ms
2021-04-16T20:29:02.790Z | LATENCY | TimeLeft : 163 | PlayerName : P041605 | Latency : 216.147995 ms | AverageLatency : 219.092331 ms
2021-04-16T20:29:02.810Z | LATENCY | TimeLeft : 163 | PlayerName : J041606 | Latency : 9.744000 ms | AverageLatency : 10.064440 ms
2021-04-16T20:29:03.807Z | LATENCY | TimeLeft : 162 | PlayerName : P041605 | Latency : 218.186005 ms | AverageLatency : 219.083267 ms
2021-04-16T20:29:03.828Z | LATENCY | TimeLeft : 162 | PlayerName : J041606 | Latency : 10.372000 ms | AverageLatency : 10.067515 ms
2021-04-16T20:29:04.823Z | LATENCY | TimeLeft : 161 | PlayerName : P041605 | Latency : 218.707993 ms | AverageLatency : 219.079514 ms
2021-04-16T20:29:04.843Z | LATENCY | TimeLeft : 161 | PlayerName : J041606 | Latency : 9.906000 ms | AverageLatency : 10.065901 ms
2021-04-16T20:29:05.833Z | LATENCY | TimeLeft : 160 | PlayerName : P041605 | Latency : 222.906006 ms | AverageLatency : 219.117783 ms
2021-04-16T20:29:05.875Z | LATENCY | TimeLeft : 160 | PlayerName : J041606 | Latency : 10.958000 ms | AverageLatency : 10.074821 ms
2021-04-16T20:29:06.614Z | SPELLCAST | SpellType : FaeWings | CasterName : J041606 | ProjectileID : 1
2021-04-16T20:29:06.657Z | SCALEHITBOX | PlayerName : J041606 | Latency : 219.117783 ms | HeadShotScaleFactor : (3.948695,3.948695,3.948695) | BodySh
2021-04-16T20:29:06.852Z | LATENCY | TimeLeft : 159 | PlayerName : P041605 | Latency : 219.998001 ms | AverageLatency : 219.126587 ms
2021-04-16T20:29:06.871Z | LATENCY | TimeLeft : 159 | PlayerName : J041606 | Latency : 9.790000 ms | AverageLatency : 10.071973 ms
2021-04-16T20:29:07.306Z | SPELLCAST | SpellType : Fireball | CasterName : P041605 | ProjectileID : 1
2021-04-16T20:29:07.862Z | LATENCY | TimeLeft : 158 | PlayerName : P041605 | Latency : 215.957993 ms | AverageLatency : 219.094894 ms
2021-04-16T20:29:07.897Z | LATENCY | TimeLeft : 158 | PlayerName : J041606 | Latency : 10.580000 ms | AverageLatency : 10.077053 ms
2021-04-16T20:29:08.039Z | SPELLRESULT | SpellType : Fireball | CasterName: P041605 | DamagedPlayerName: InstancedFoliageActor_0 | ProjectileID : 1 |
2021-04-16T20:29:08.872Z | LATENCY | TimeLeft : 157 | PlayerName : P041605 | Latency : 223.259995 ms | AverageLatency : 219.136551 ms
2021-04-16T20:29:08.889Z | LATENCY | TimeLeft : 157 | PlayerName : J041606 | Latency : 10.162000 ms | AverageLatency : 10.077903 ms
2021-04-16T20:29:08.997Z | SPELLCAST | SpellType : Fireball | CasterName : P041605 | ProjectileID : 2
2021-04-16T20:29:09.558Z | SPELLRESULT | SpellType : Fireball | CasterName: P041605 | DamagedPlayerName: wLandscape | ProjectileID : 2 | Hit : 0 | He
2021-04-16T20:29:09.876Z | LATENCY | TimeLeft : 156 | PlayerName : P041605 | Latency : 215.981995 ms | AverageLatency : 219.105011 ms
2021-04-16T20:29:09.909Z | LATENCY | TimeLeft : 156 | PlayerName : J041606 | Latency : 9.936000 ms | AverageLatency : 10.076484 ms
2021-04-16T20:29:10.887Z | LATENCY | TimeLeft : 155 | PlayerName : P041605 | Latency : 218.276001 ms | AverageLatency : 219.096710 ms
2021-04-16T20:29:10.919Z | LATENCY | TimeLeft : 155 | PlayerName : J041606 | Latency : 10.062000 ms | AverageLatency : 10.076340 ms
2021-04-16T20:29:11.902Z | LATENCY | TimeLeft : 154 | PlayerName : P041605 | Latency : 218.162003 ms | AverageLatency : 219.087372 ms
2021-04-16T20:29:11.919Z | LATENCY | TimeLeft : 154 | PlayerName : J041606 | Latency : 11.048000 ms | AverageLatency : 10.086057 ms
2021-04-16T20:29:12.913Z | LATENCY | TimeLeft : 153 | PlayerName : P041605 | Latency : 223.145996 ms | AverageLatency : 219.127960 ms
2021-04-16T20:29:12.939Z | LATENCY | TimeLeft : 153 | PlayerName : J041606 | Latency : 10.212000 ms | AverageLatency : 10.087316 ms
2021-04-16T20:29:13.676Z | FLAGPICKUP | Name : J041606 | Flag : Blue
2021-04-16T20:29:13.927Z | LATENCY | TimeLeft : 152 | PlayerName : P041605 | Latency : 214.065994 ms | AverageLatency : 219.077332 ms
2021-04-16T20:29:13.942Z | LATENCY | TimeLeft : 152 | PlayerName : J041606 | Latency : 9.768000 ms | AverageLatency : 10.084123 ms
2021-04-16T20:29:14.948Z | LATENCY | TimeLeft : 151 | PlayerName : P041605 | Latency : 219.570007 ms | AverageLatency : 219.082260 ms
2021-04-16T20:29:14.973Z | LATENCY | TimeLeft : 151 | PlayerName : J041606 | Latency : 10.772000 ms | AverageLatency : 10.091002 ms
2021-04-16T20:29:15.964Z | LATENCY | TimeLeft : 150 | PlayerName : P041605 | Latency : 216.606003 ms | AverageLatency : 219.057495 ms
2021-04-16T20:29:15.985Z | LATENCY | TimeLeft : 150 | PlayerName : J041606 | Latency : 9.874000 ms | AverageLatency : 10.088831 ms
2021-04-16T20:29:16.664Z | SCALEHITBOX | PlayerName : J041606 | Latency : 219.057495 ms | HeadShotScaleFactor : (3.948280,3.948280,3.948280) | BodySh
2021-04-16T20:29:16.974Z | LATENCY | TimeLeft : 149 | PlayerName : P041605 | Latency : 221.472000 ms | AverageLatency : 219.081650 ms
2021-04-16T20:29:17.006Z | LATENCY | TimeLeft : 149 | PlayerName : J041606 | Latency : 9.914000 ms | AverageLatency : 10.087083 ms
```

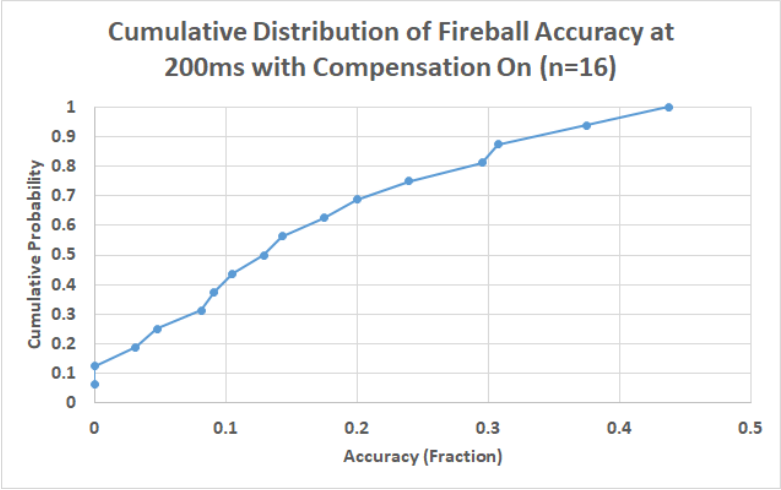
Appendix 5.A: Cumulative Distributions of Fireball Accuracy

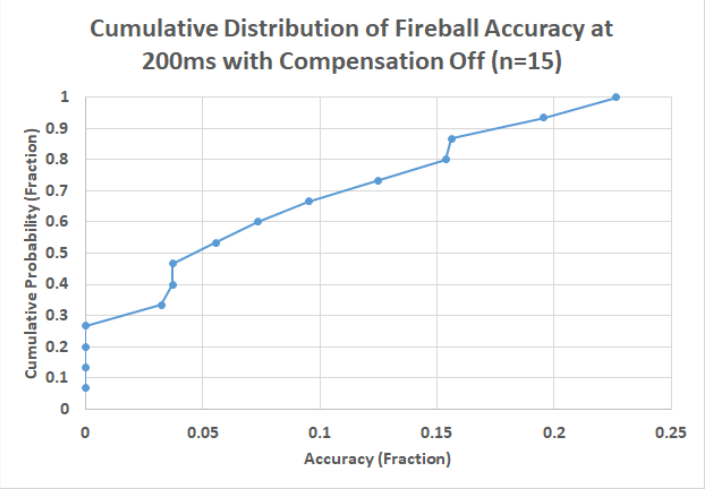




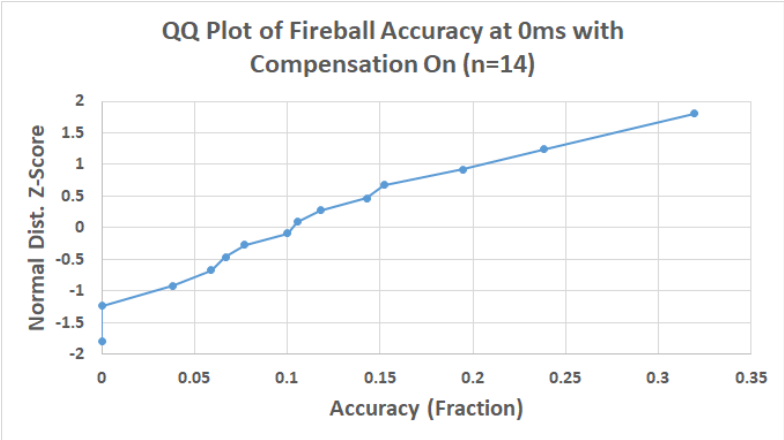


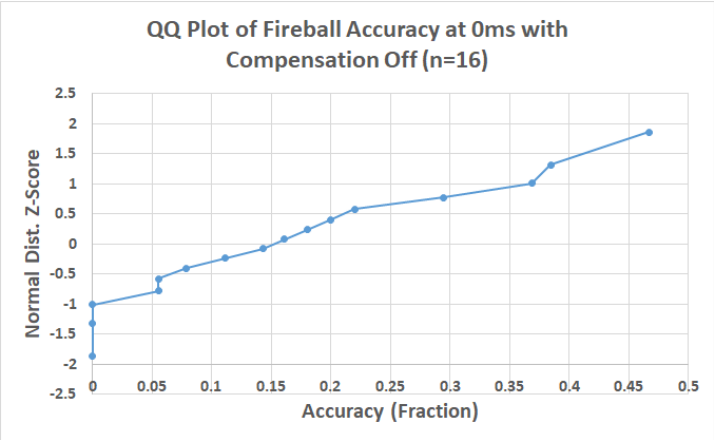


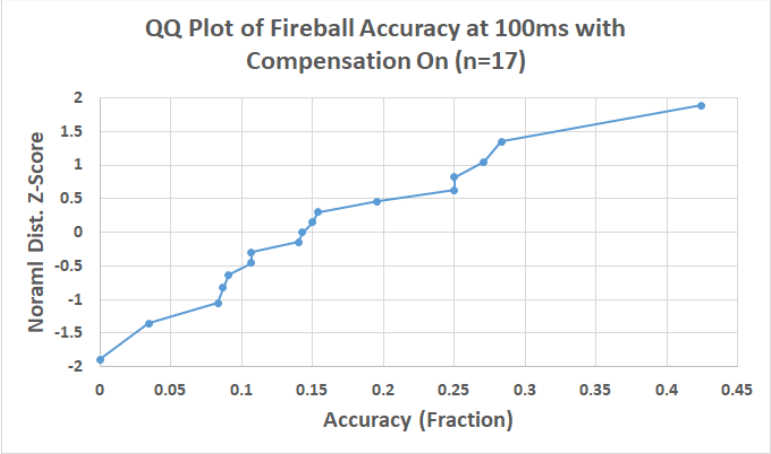


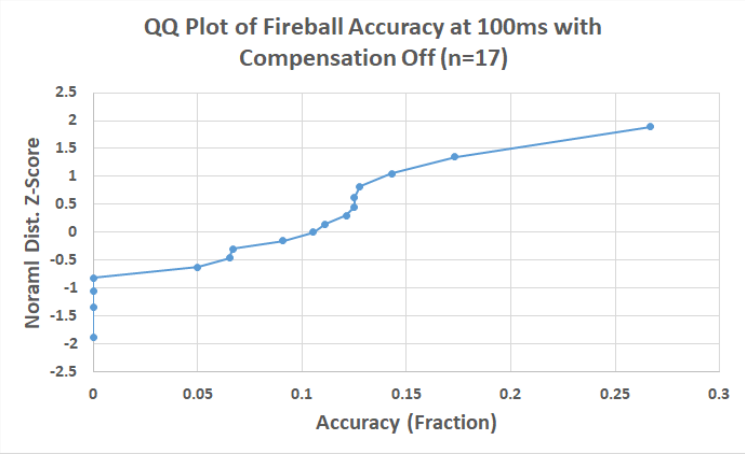


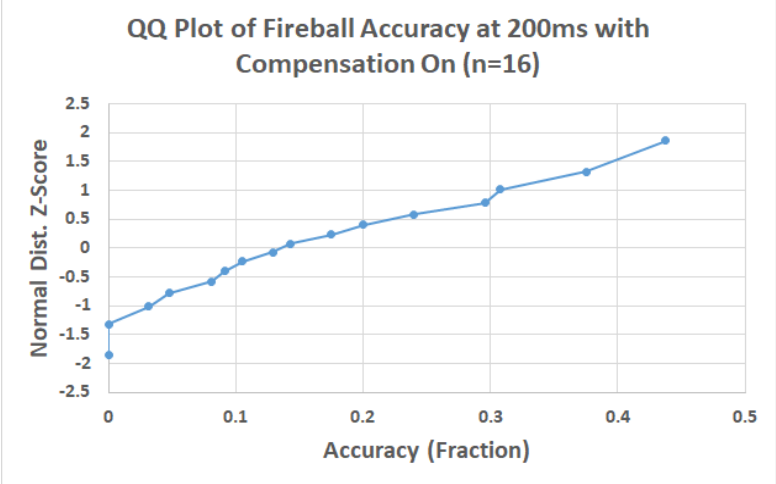
Appendix 5.B: Quantile-Quantile Plots of Fireball Accuracy

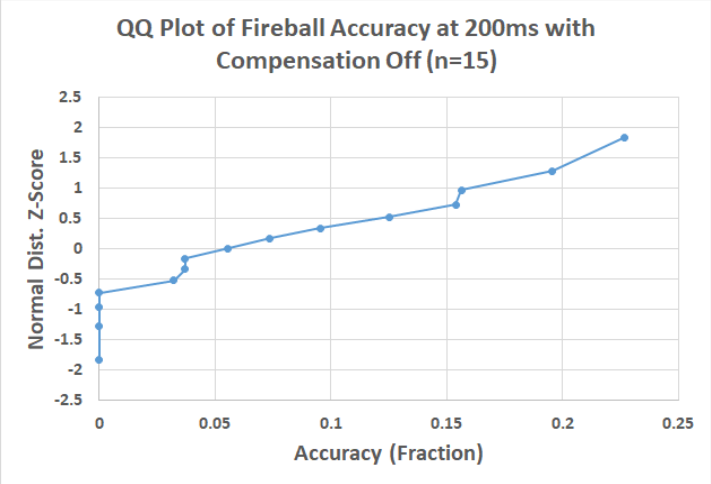




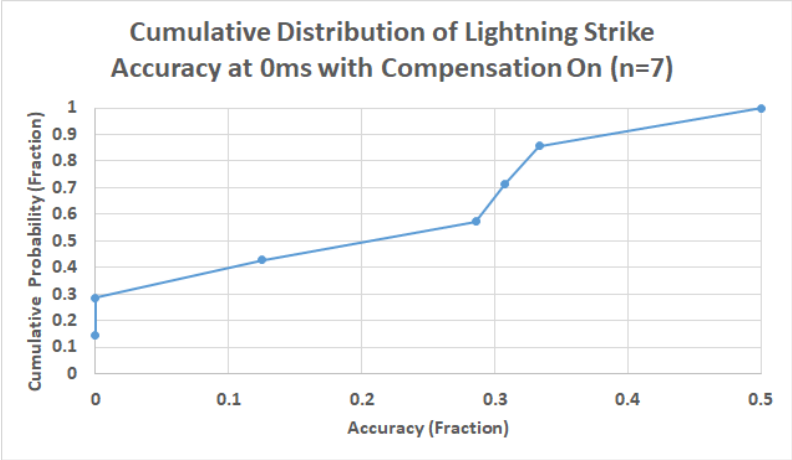


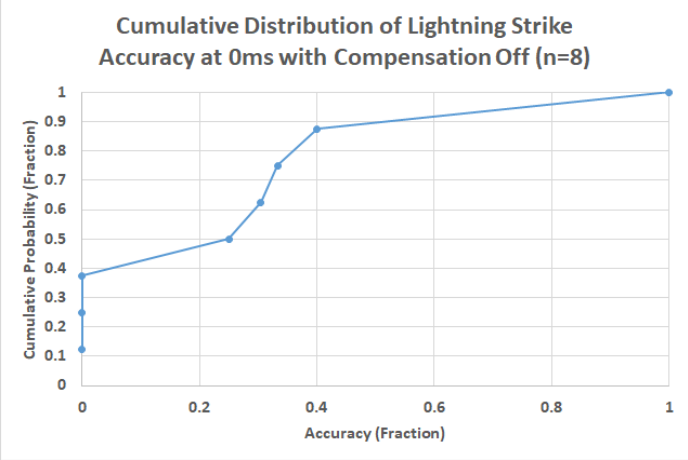


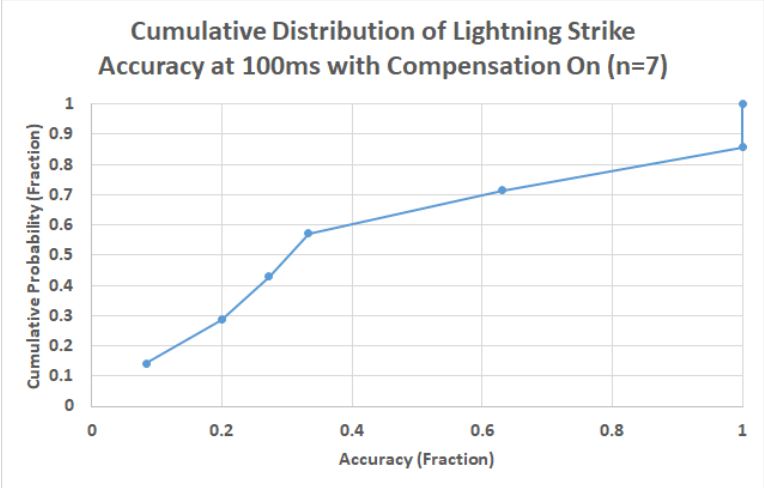


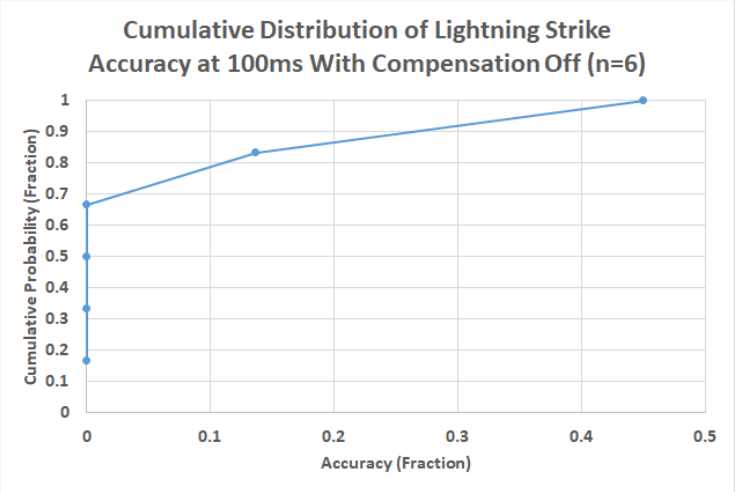


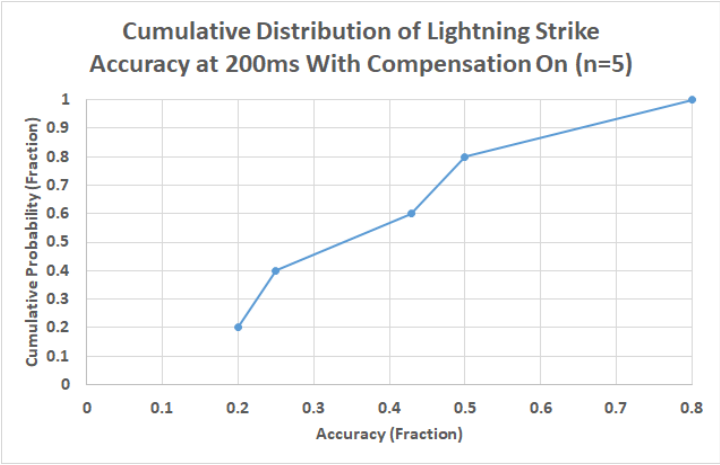
Appendix 5.C: Cumulative Distributions of Lightning Strike Accuracy

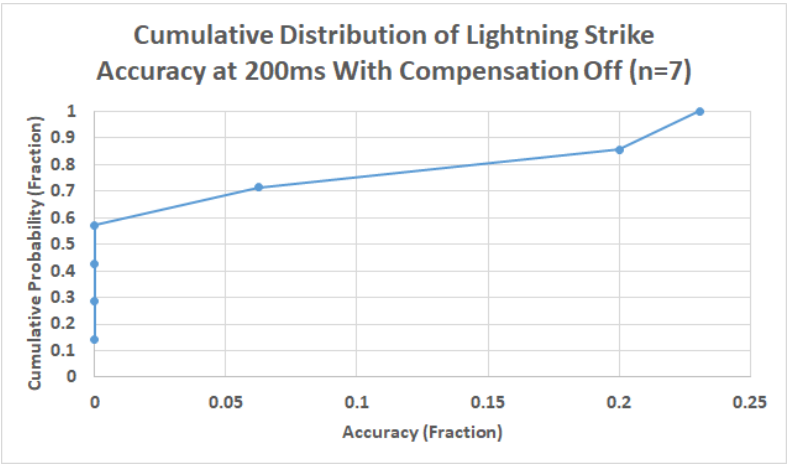




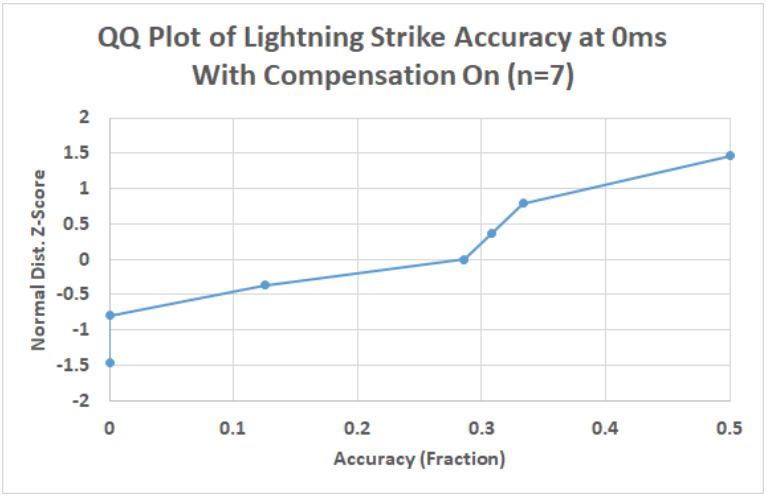


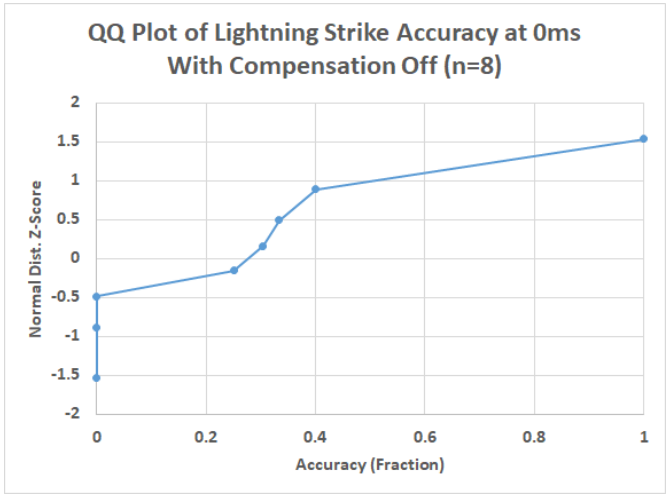


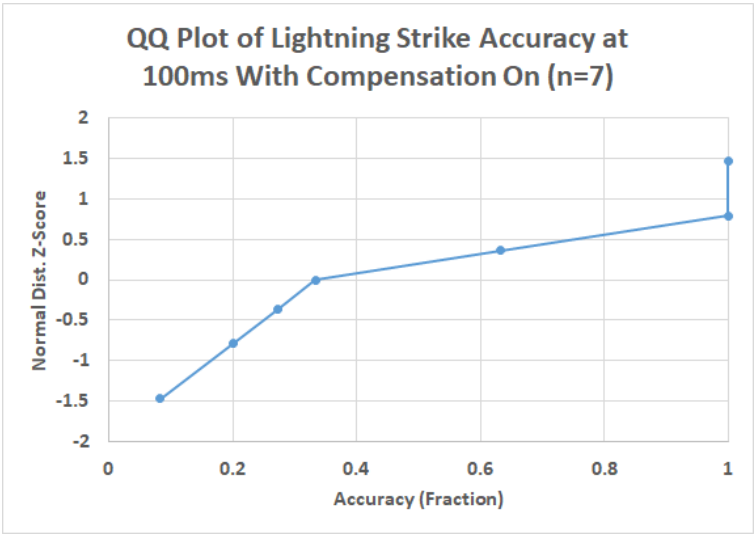


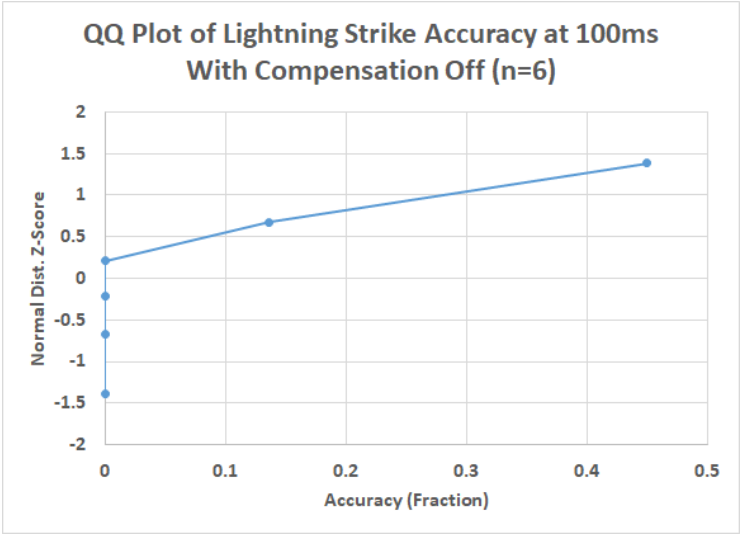


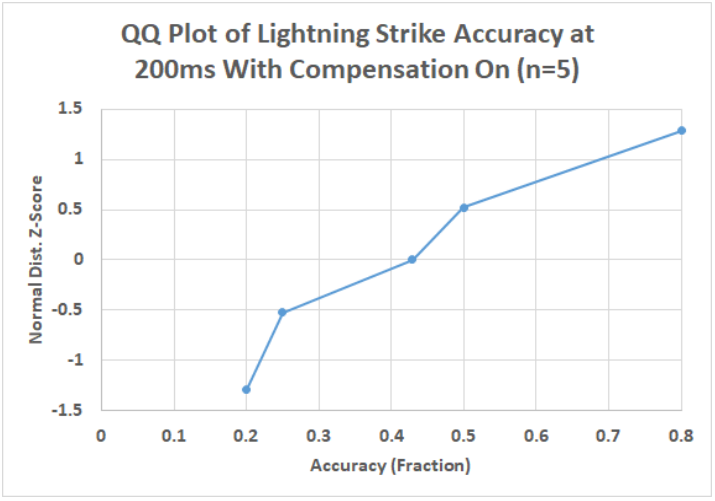
Appendix 5.D: Quantile-Quantile Plots of Lightning Strike Accuracy

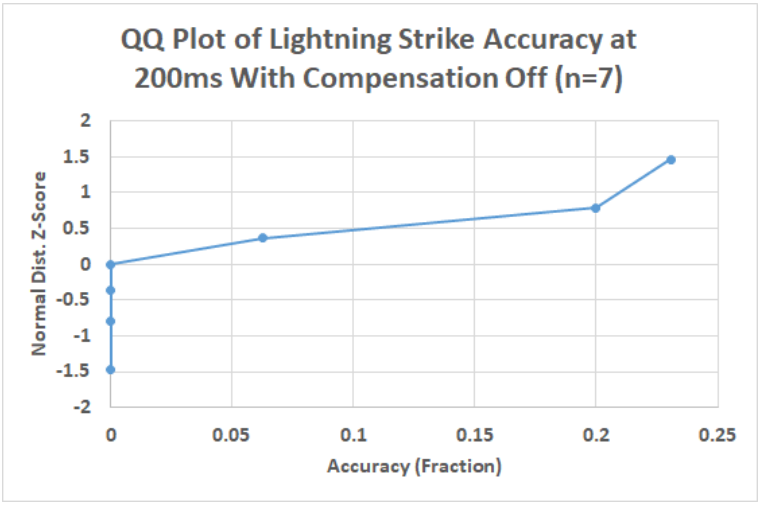






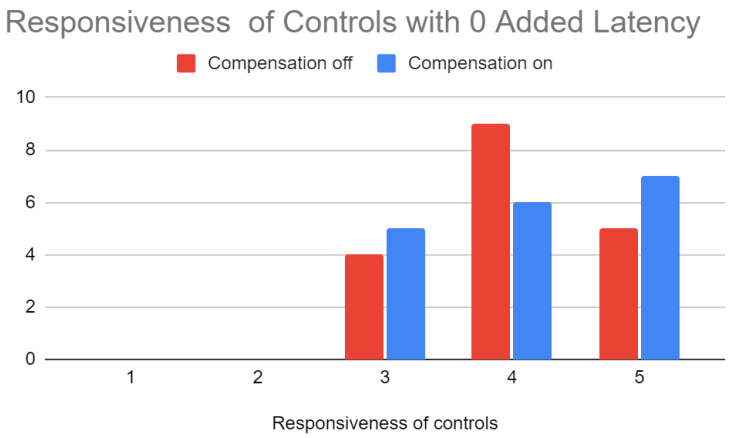


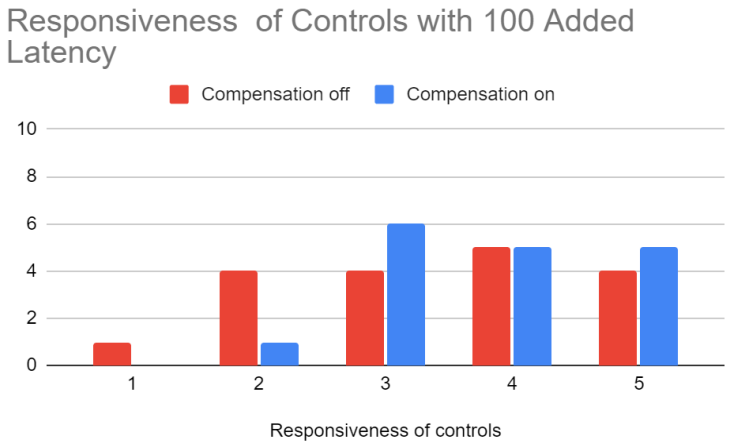


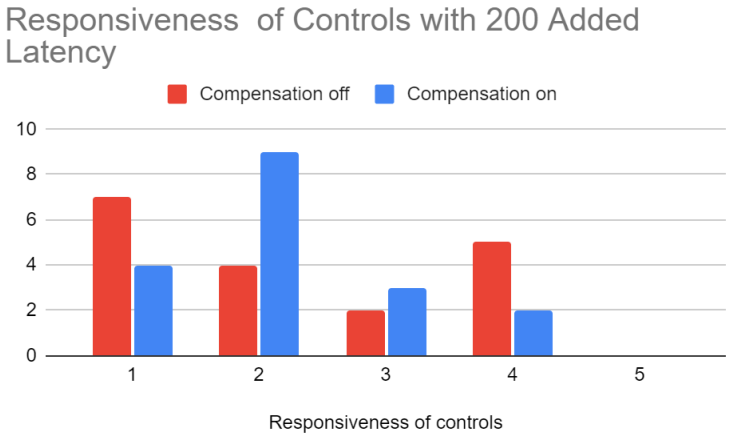


Appendix 5.E: Additional QoE questions in Post-trial Surveys

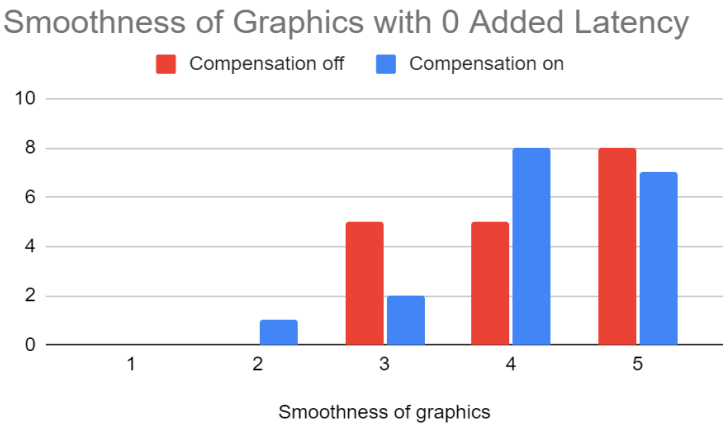
Rate the responsiveness of the controls (1 - Low, 5 - High):

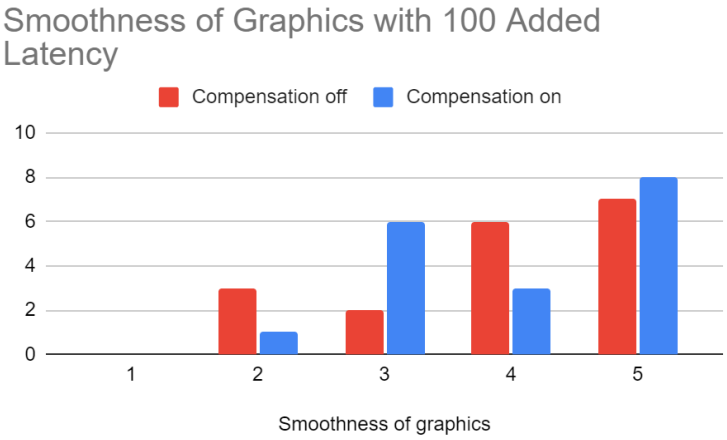


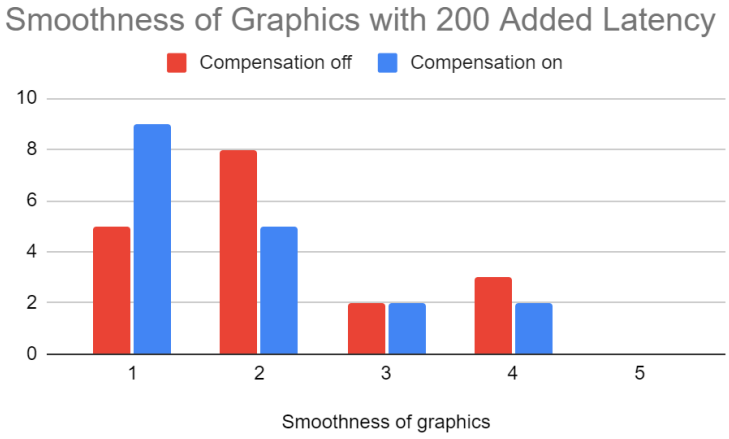




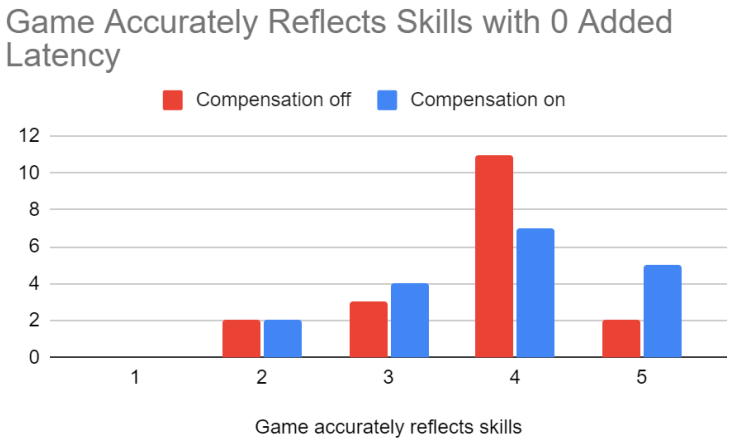
Rate the smoothness of the graphics (1 - Low, 5 - High):

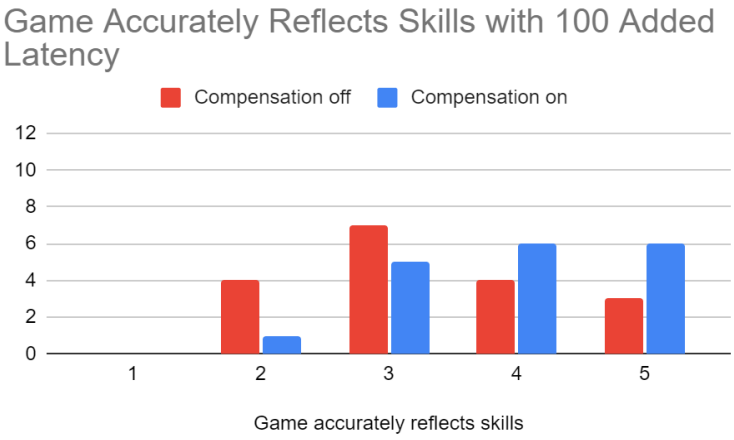


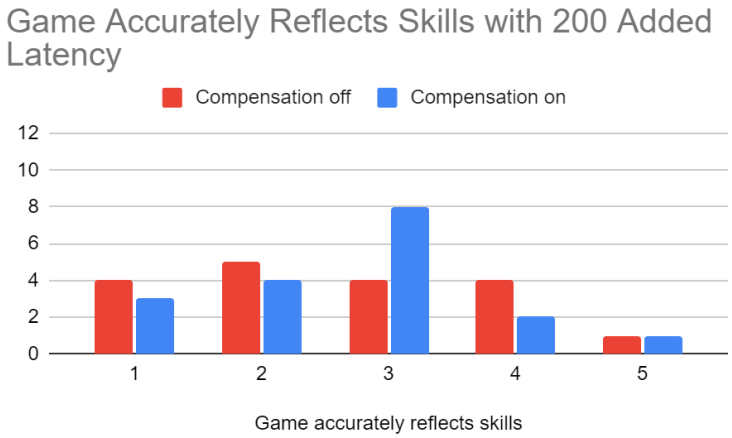




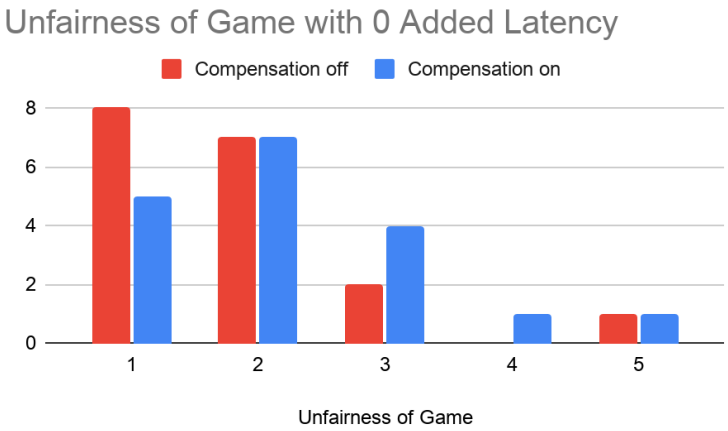
Rate how well you felt your performance in the game accurately reflected your skill (1-Low, 5 - High):

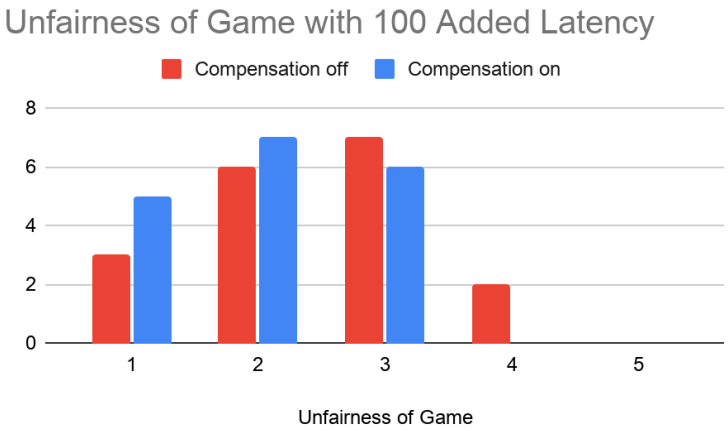


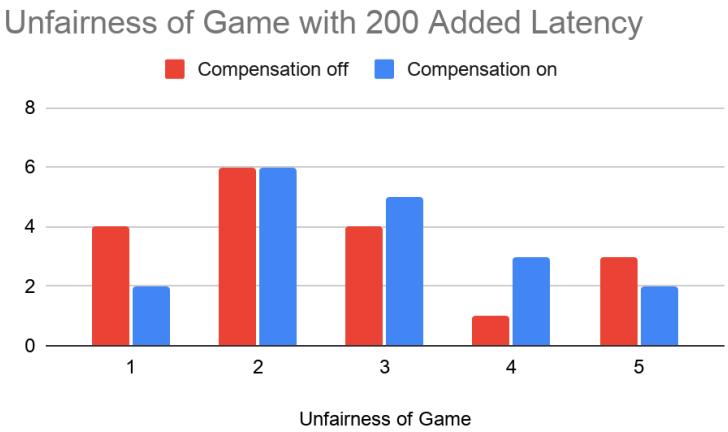




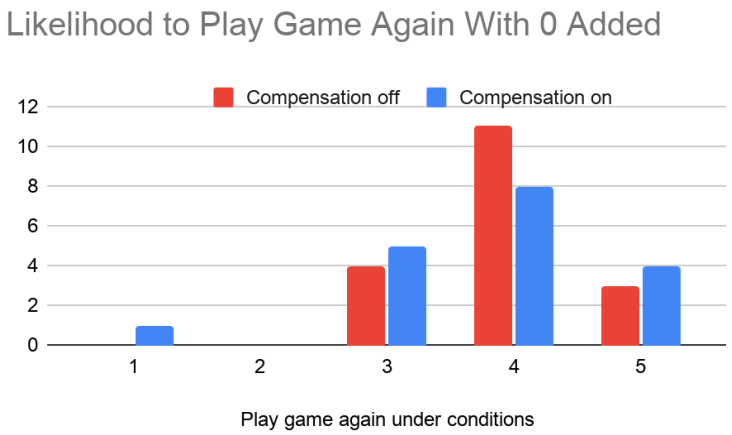
Rate how unfair the game felt (1-Low, 5 High):

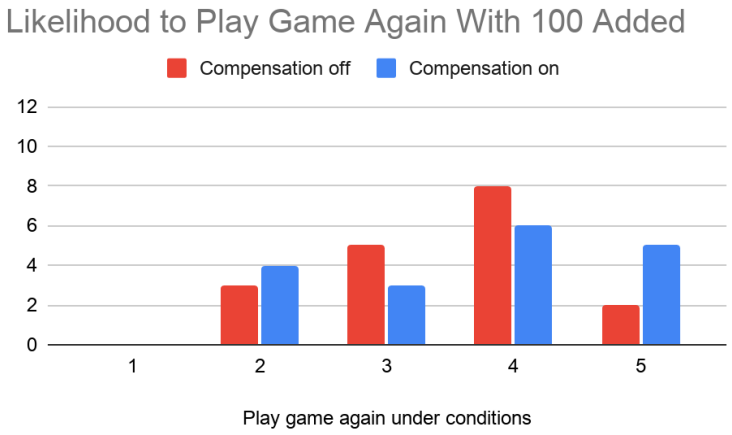


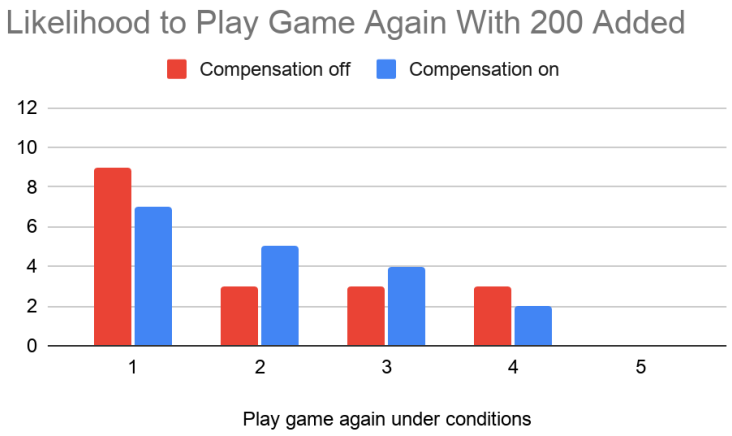




Rate your likelihood to play the game again under these conditions (1-Low, 5 -High):







Appendix 5.F: Gameplay Survey Feedback

	Responses
	your game is fun! good luck on the data analysis and everything
	Game definitely makes more sense the more you play. Still had trouble knowing if I hit my targets. The flame tornado felt a bit wonky when you got hit by it but it's a cool idea. A bit rough around the edges but I'd definitely give it another try.
	map design was barren
	Include instructions before the game is played, the goal, the controls, what each element does and it will be better. Very fun! Thank you!
	Not really, good job overall.
	Just make everyone a hair squishier.
	In the middle the bots seemed not amazing (to me) and then in the last round the bots got really smart, though I don't know how well that correlates to an average player when my metrics are competitive FPS players with whom I frequently play.
	The AI was impressively competent and did use a variety of abilities, but the tendency to stop moving when casting and the obvious inconsistent path planning made them very distinguishable from a human player. The number of spells seemed quite high. I'm not sure I was able to use all of them in a single match and tended to rely on 2-3 at a time. (Making the tab menu accessible on the waiting for match screen may have helped)
	Such a great game, nice job!!
	There were a lot of spells, also it was hard to kill anybody when they had the flag meaning most games were a tie
	The combat of the game was difficult to understand without taking the time to carefully read through the spells and get a sense of how they work.