Attribute Scaling for Latency Compensation in Cloud Games

Edward Carlson, Tian Yu Fan, Zijian Guan

March 2020

An Interactive Qualifying Project Report:
submitted to the Faculty of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the
Degree of Bachelor of Science

Approved By:
Professor Mark Claypool, Advisor

Abstract

Thin clients have become increasingly popular in the gaming industry. This architecture alleviates the computational burden on local hardware but is susceptible to latency as it requires round-trip communication from client to server for all player input. To reduce the effects of latency, attribute scaling has the potential to boost player performance. We conducted user studies to determine how different latencies and attribute scaling techniques affect game performance and enjoyability. We found that increases in added latency significantly decrease user performance and enjoyability. Furthermore, increasing attribute scaling significantly increases user performance, indicating that it may be an effective compensation method for latency. Furthermore, we developed a mathematical model to determine how much games should compensate given the latency, difficulty, and desired accuracy.

Table of Contents

Section 1. Introduction

Interest in thin clients has become increasingly popular in the gaming industry. For example, the current thin client market is valued at over a billion dollars, with it expected to grow by over 15% in the next decade [1]. In the past four years, major companies such as Google, Microsoft, and Electronic Arts have announced the development of their thin client platforms, where previously thin clients were almost unheard of [2].

Traditional gaming consoles are thick clients as most data processing is completed on the local machine, relying on frequent maintenance from the user to keep up with the newest software and hardware requirements of modern games. Conversely, thin clients process data on a server. The client requests a game state update or signals a user input to the server. Then, the server performs the appropriate tasks based on the request and returns feedback in the form of a video stream.

Thin client architectures eliminate the need for traditional gaming consoles and expensive hardware in homes as datacenters take over game processing responsibilities. This enables gamers to play across multiple platforms, including desktop, tablet, and phone, without having to install or patch. Games can become more shareable and accessible across the Internet, allowing streamers to reach a wider audience and viewers to consume their desired content more conveniently. Developers are freed from the limitations of specific consoles since they can take advantage of the advanced hardware in the data centers.

Despite these promising claims, however, thin clients are vulnerable to latency, which is the time delay between player input and feedback from the system [3]. Thick clients process game data on the local hardware, resulting in quick feedback. However, thin clients must communicate extensively with data centers for processing, resulting in additional latency from the network. Traditional online games support low-bitrate streams by sending small commands, preventing over-consumption of bandwidth. However, cloud games operate on high-quality video streams, requiring high bitrates that could overwhelm available capacities. Network congestion and packet loss can delay data transmission, resulting in latency and decreased responsiveness of the game. When latency is high, the player experiences laggy controls, degraded visual quality, and a less responsive and rewarding experience [4]. Significant latency

can weaken player performance and increase their frustrations, leaving them with unpleasant experiences with thin clients.

To address this issue, one type of latency compensation method provides the opportunity for performance improvement proportional to the added latency by scaling the attribute of select game elements in response to latency. This technique is to be integrated into these games from our user studies. The first game uses spatial compensation where the size of the target hitbox enlarges as latency does. The second game uses temporal compensation where the hitbox duration increases proportionally to the latency. While under-compensation could frustrate the player, overcompensation could make the game too easy. Game developers must find a balance between these two extremes.

The purpose of our study is to construct a model which outputs an appropriate compensation factor such that the game rewards the player for their performance while preserving the intended difficulty. Our research uses two games developed by students at the Worcester Polytechnic Institute (WPI). The first game, titled Catalyst, is a first-person shooter where a target runs back-and-forth across a platform. Latency affects mouse input and visual feedback, difficulty controls the speed at which the target travels, and the compensation factor determines the hitbox size of the target. The second game, titled Nova, is a rhythm-based game where notes spawn on the screen based on the beat of a song. Latency affects mouse input and visual feedback, difficulty controls the rate at which new targets are made, and the compensation factor determines the time at which the note remains on the screen. Participants of the user study played these games in random order where each round has a different random combination of latency, difficulty, and compensation factors. After each round, their performance was recorded and their enjoyment was evaluated through a survey.

Twenty-four participants engaged in our user study, each of whom played both games with rounds at varying latencies, difficulties, and compensation factors. The latencies which were tested were: 0ms, 50ms, 100ms, 125ms. The difficulties for Catalyst which controlled the target travel speed were 200cm/s (easy) and 500cm/s (hard). The difficulties for Noa which controlled the note cooldown time were 1s (easy) and 0.5s (hard). The compensation factors, which were multiplied with the default no-latency value, were 1.0, 1.5, and 2.0.

In our analysis, we observed that increases in latency significantly decrease user performance and enjoyability. Furthermore, increasing attribute scaling significantly increases

user performance, indicating that may be an effective compensation method for latency. Using this collected data and analysis, we derived mathematical models which could help determine the appropriate attribute scaling amount for games based on added latency.

The remainder of this report provides detailed information on our research. Section 2 compiles background information related to our work, Section 3 explains our methodology for the user studies, Section 4 presents an in-depth analysis of the collected data, Section 5 summarizes our findings, and Section 6 offers suggestions for future research.

Section 2. Background Research

This chapter addresses some background information related to our research. Specifically, we discuss various network architectures, sources from which latency arises, compensation techniques, measurement metrics, game actions, and methodologies from existing research.

Section 2.1 Network Architectures

Thick and thin clients are the main network architectures in the gaming industry. Thick clients depend on local hardware for computational power, while thin clients communicate with external servers for such information. This section discusses the advantages and disadvantages of each architecture.

Section 2.1.1 Thick Client Architecture

Most gaming hardware, such as personal computers (PCs) and consoles, are designed around thick clients. This architecture has most of its essential resources and applications installed onto the local hard drive [5]. This allows the client to process data and render graphics locally rather than over a network, reducing network latency and reliance on connectivity. While local latency remains present, it is significantly reduced because not all data needs to travel beyond the local computer, resulting in faster feedback upon user input and smoother gameplay. Nevertheless, multiplayer games running on thick clients still require continuous game state updates from the server, so stable connectivity is needed regardless of the architecture. Thick clients are configurable because users can customize their hardware and software preferences. However, this freedom could become a financial burden as hardware must be frequently updated to meet the game's system requirements.

Section 2.1.2 Thin Client Architecture

Recently, leading technology companies, like Google, have shown interest in delivering games on a thin client architecture. Unlike thick clients, a thin client architecture delegates the majority of its computational work to centralized servers [6]. First, the client requests a game state update or signals a user input to the server. Then, the server performs the appropriate tasks

based on the request and returns feedback in the form of a video stream. Finally, the client displays new graphics and plays audio. This reliance on continuous network communication means latency may result in unpleasant gameplay. Our research involves testing games while simulating the effects of latency to determine how they affect user performance and experience.

Thin clients require cloud providers to engineer hardware capable of sustaining the required power and network capacity, as well as spreading data centers across the globe to minimize latency. Hardware maintenance becomes easier for regular gamers since their local machines do not have to perform the processing work. Thin clients also eliminate the necessity for local updates, downloads, and patches since these changes are managed by the data centers as well. Game developers have more design freedom if they take advantage of the powerful hardware from these data centers [7]. Finally, thin clients enable streaming across a variety of platforms, including desktops, laptops, tablets, and phones.

Section 2.2 Latency sources

Latency is the time delay between user input and the output returned by the server. The main sources of latency in a thin client architecture include network delay, processing delay, and playout delay [8].

Section 2.2.1 Network delay

Network delay arises from the communication between client and server. This communication can be hindered by the available bandwidth, which is the maximum rate of information that can be transferred [9]. Traditional online games transmit data frequently at low bitrates to conserve bandwidth [3]. However, low bitrates are difficult to achieve for games designed for thick clients because they generally involve more sophisticated controls, graphics, and mechanics. As a result, they must transfer large amounts of data which can congest the network.

The physical distance between the client and server can also contribute to increased latency. The data must travel farther for a player who is located across the globe than someone who is near the server center. This issue generates complications in multiplayer games if the game state for each player fails to update simultaneously, desynchronizing the gameplay.

Different connectivity methods can impact the amount of experienced latency. According to a previous study, local area networks and broadband access networks can add tens of milliseconds to latency [10]. The Internet can add fifty milliseconds depending on distance, and cable and dialup modems can add hundreds of milliseconds.

Network congestion occurs when the network manages more data than it is capable of handling and is another source of latency. This results in the mishandling of packets, or small groupings of data. Packets can be delayed, lost, corrupted, reordered, or duplicated, which can produce unexpected behavior during gameplay [11]. The application could also time out and disconnect because the anticipated data took too long to arrive.

The aforementioned sources of network delay are simulated during our experiment to determine the effects of latency on user performance and experience.

Section 2.2.2 Processing Delay

Processing delay in a cloud gaming server architecture is the time it takes for the server to process the player's input, render the new frame, and start sending the video stream back to the client [8]. The raw video streams which are produced by cloud games can be big, consume excessive bandwidth, and cause network congestion, as well as being more expensive to transfer. In most cases, when a video file is large, it is compressed or encoded into a smaller form using video codecs to make it quicker to download. However, trying to compress raw video streams with video codecs is often time-consuming, and cannot be done with a real-time game without adding delay.

Section 2.2.3 Playout Delay

While small, another source of delay is playout delay, which is the time it takes for the video information to be processed by the user device and displayed onto the screen. This is often not an issue for casual players, with it only being a potential problem on outdated hardware or some mobile devices, or if the device is already under considerable load. However, playout latency could be detrimental to competitive gamers who require fast processing feedback [12]. We will be simulating playout delay during our research to determine how delaying the response to user input affects player performance and experience.

Section 2.2.4 Cloud-Based Delay

Unlike traditional gaming frameworks, cloud gaming relies on frequent data exchange with remote servers which makes them susceptible to cloud-based latencies. This genre of latency is a combination of network, processing, and playout delays [8]. Since most of the information processing is performed on the remote servers, more frequent communication at higher bandwidths is required, increasing vulnerability to network delay. This also increases processing and playout delays as the user inputs and video information must travel between the local hardware and the server to be registered.

While the measurement of network delays can be done quantitatively using pings, the processing and playout delays are difficult to measure because these delays occur internally in the cloud gaming system, and the fact that most of the cloud gaming services are proprietary software instead of being open-source adds to that difficulty of measuring delays precisely, but researchers are trying to get around those difficulties by isolating singular event (invoking menu) and decomposing three parts of the delay [8].

Section 2.3 Existing Compensation Techniques

To reduce the effects of latency on gameplay, games that rely on external servers use compensation techniques. Example compensation techniques include geometric compensation.

Section 2.3.1 Geometric Compensation

Geometric compensation adjusts the physical components of the game to compensate for the effects of latency [14]. In a previous study, Flappy Bird was used to test this method. The goal of this game is to control a flying bird through small gaps between vertical pillars. The bird flaps upward when the user clicks on the screen. If the button is not pressed, the bird falls downward. If the bird collides into a pillar, the game is over and the player has lost. Under the effects of latency, user click input to the server could be delayed or entirely lost. As a result, the player could lose even though the bird would have successfully cleared the pillar in an ideal condition without latency. To compensate for this issue, the hitbox of the pillars can be reduced depending on the amount of latency. This method has been shown to reduce the failure rate of

high-latency players. However, its potential might be limited to games that have relatively simple tasks. Furthermore, it could affect the player's quality of experience in unexpected ways.

The two games which we tested have compensation techniques implemented. The first game, a first-person shooter, experiments with geometric compensation, where the hitbox of the target increases as the latency increases. The second, a rhythm game, increases the amount of time for which hitboxes are present on the screen as latency increases.

Section 2.4 Precision and Deadline

Claypool & Claypool [3] introduces a novel way to classify the effects of latency on game actions. These new measurement metrics, deadline, and precision allow us to measure attributes of game actions quantitatively. In general, game actions which require high precision and tight deadlines are more affected by latency than those which require low precision and loose deadlines. This suggests that compensation models work on a case-by-case basis as deadline and precision are different for each game action.

Section 2.4.1 Precision

Precision is the accuracy required for the action to be completed successfully. For example, shooting a target with a sniper rifle requires significantly more precision than a drag and drop motion when moving things around in an inventory. The distance between the center of the scope and the target hitbox must be very small for the shooting action to be successful. When dragging and dropping in the inventory, the hitboxes are typically larger, increasing the chances of a successful action.

Section 2.4.2 Deadline

Deadline is the time by which the action must be completed. The research provides the example of walking across a suspended beam. If the suspended beam is straight, then the player would have good intuition on how to complete the task. However, if the beam is segmented into smaller twisting parts, then the player would need more frequent feedback from the server about their state and location. An increase in latency would increase the likelihood of the task failing as

the feedback is delayed. In Nova, we change the duration at which the target remains on the screen based on latency, adjusting the deadline.

Section 2.5 Game Actions

Modern gaming combines numerous mechanics, each requiring different methods of compensation. The selected games from our user study focus on shooting and timing actions. Table 1 describes popular actions found in video games.

Table1. Common actions in video games

Action	Description	Example
Pointing	Adjusting the marker onto a target	Aiming at an enemy using a sniper rifle in FPS
Tracking	Keeping aim on the target	Keeping a crosshair on the target to maximize damage
Shooting	Releasing a trigger	Shooting a rifle to harm an enemy
Reaction	Suddenly reacting to an event	Defense moves to block attacks in Street Fighter
Timing	Acting at a specific time	Jumping onto moving platforms
Selection	Clicking on an item	Choosing an item in an inventory
Navigation	Navigating a character around the world	Running in the world map in Zelda

Section 2.6 Previous Research Methodology

To familiarize ourselves with common procedures and tools used in this field, we researched methodologies from previous studies related to the effects of latency on games.

Section 2.6.1 Methods for Selecting Game

Game selection is one of the most important steps in testing the effects of latency, and so different studies use different games for different reasons. For example, in Desveaux et al's [14] paper on differences between different cloud gaming providers, they used the game Assassin's Creed Odyssey, as it was on a variety of different providers. In another paper [15], the game Neverball was used as it was a game that most people had not played, and they wanted to make sure player skill did not affect the results. In the study of geometric compensation [13], the researchers use Flappy bird because its mechanic is relatively simple; it is exclusively timing action, so the researchers could narrow down their way and facilitate a focused mathematical model. Also, the game can be easily modified or rebuilt for research purposes. The games for our research will be provided to us by two Major Qualifying Project teams at the Worcester Polytechnic Institute.

Section 2.6.2 Measuring Metrics

Often, the best way to measure the user's quality of experience is to simply ask them how they felt with a survey. In Long et al's paper [4], they asked the users multiple questions on how the task felt with multiple different latencies, and the trend was that as latency increased, frustration increased while the enjoyment and perceived responsiveness fell.

While the question of how latency affects the quality of experience is a subjective one, there are some objective measures of the effect of latency through analyzing play performance. One way to measure the effect of latency objectively is to track how many times a user misclicks between different amounts of latency. Another is how often a user's cursor leaves the optimal path it could take. In Long et al's paper [4], performance with these metrics saw a sharp decline with users that used a mouse, showing that objective measures of performance agree with results from surveys and other more subjective measures of quality of experience. However, one of the major problems with this method of measuring the objective quality of experience is that it requires specialized software to track user input.

During our experiment, we measured hit percentage, which is the number of successful hits on the target divided by the total number of attempted shots, for both games. We will also be asking the participants survey questions after each trial. These include:

1. How enjoyable was the session? (Rate from 1-5)

2. How frustrating was the task? (Rate from 1-5)
3. How challenging was the last session? (Rate from 1-5)
4. How well do you think you performed? (Rate from 1-5)
5. How responsive were the controls? (Rate from 1-5)
6. Based on the quality of experience, how likely are you willing to play the game again? (Rate from 1-4)

Section 2.6.3 Demographics

For each participant, it is important to collect their demographic information because it provides context to the research and could reveal insights on any biases during the experimentation. Similar studies have asked for the user's age, gender, major, whether they have previously used a cloud-based game service, whether they have played the selected game previously, their preferred game genres, their self-rating as a gamer, and the number of hours of games they play over a week. We asked:

1. Age
2. Major (s)
3. Gender
4. How many hours on average do you spend playing games in a typical week?
5. Have you played games on a cloud-based game service before?
6. Which device do you typically play on?
7. How would you rate yourself as a gamer on a PC?
8. How would you rate yourself as a gamer on a console?
9. How would you rate yourself in an FPS game?
10. How would you rate yourself in a Rhythm game?
11. What kind of games do you typically play?

Section 2.7 Summary

In this chapter, we discussed thick and thin client architecture, various sources of latency, existing compensation techniques, measurement metrics, game actions, and methodologies. In our study, shooting and timing actions were tested against progressively incremental levels of latency to determine how it affects user performance and experience. Furthermore, we

established a mathematical model which determines the amount of compensation needed based on the added latency. We hope our study could offer some empirical references and insights for the developers of games and cloud gaming platforms, so they could minimize the negative effect of latency for the cloud gaming experience.

Section 3 Methodology

To determine the effects of latency and attribute scaling on user performance and quality of experience, our team developed user studies where human participants played two games. These games, each with different core actions, were tested under varying latency, difficulty, and attribute scaling. This section describes the methods used to select the games, recruit participants, develop tools, conduct the study, and accommodate for the COVID-19 pandemic.

Section 3.1 Game Selection

Games with different core actions were selected to determine their performance and the effectiveness of their attribute scaling under varying latency. This section describes the criteria of the games which were under consideration. Furthermore, it introduces the selected games which were included in the final version of our study.

Section 3.1.1 Selection Criteria

The games which were considered must have met the following criteria:

1. The game has an action in which the player score can be displayed or recorded. This ensures that the player performance is measurable and analyzable.
2. The studied game action must be isolated as it encourages the player to focus on the specific task and reduces the number of confounding factors in the study.
3. The game is susceptible to artificial latency. To fulfill the purpose of this research, we must control input latency.
4. The studied game action must be quickly accessible within the game. To collect meaningful data, time must not be wasted setting up the game after each round.
5. The game action must be repeatable because the study must collect data under different conditions.
6. The game can be paused between rounds, allowing the participant to fill out the quality of experience survey.

Section 3.1.2 Selections

Our team considered many options when selecting the appropriate games for our study. We considered games available in the Google Stadia libraries and demo games which were found on the Internet. We ultimately decided to conduct our study using two games developed by Major Qualifying Project (senior capstone) teams at the Worcester Polytechnic Institute titled Catalyst and Nova.

Since both MQP teams are advised at our university, we were able to contact them regarding making modifications to their games for our user study. This communication was beneficial as the teams were able to design a version of their game specifically to run our user studies.

From this collaboration, we were able to control the time of each round, the independent variables (latency, attribute scaling, and difficulty), and retrieve player performance effectively. To change the latency, we passed a parameter into a configuration file.

The games had very simple mechanics which reduced the effects of confounding factors in our results. For example, in a game like Assassin's Creed Odyssey, there are multiple skills and enemies in a fighting sequence. This variability is difficult to control and could change the outcomes of player performance drastically. With simple game actions, such as the ones used in this study, we had a more controlled environment.

Since the games were developed by students, they did not have a dedicated quality assurance team. Therefore we had to cooperate with the MQP teams to make sure any performance changes were not caused by bugs.

Section 3.1.3 Overview of Games

Both games were developed in Unreal Engine 4.25. They each implement targets which the participant must hit. The score which indicates player performance is displayed on the screen. Each game focuses on one straightforward mechanic, so it is easily understood by the player and there are fewer interfering elements from the game environment. There are no extraneous features to the gameplay, so the studied action is quickly accessible. Each level is separated by a break screen which prompts the participant to fill out a quality of experience survey. Each level of the games is also controlled by a unique combination of three independent variables. These variables are latency, difficulty, and attribute scaling.

Section 3.1.3.1 Catalyst

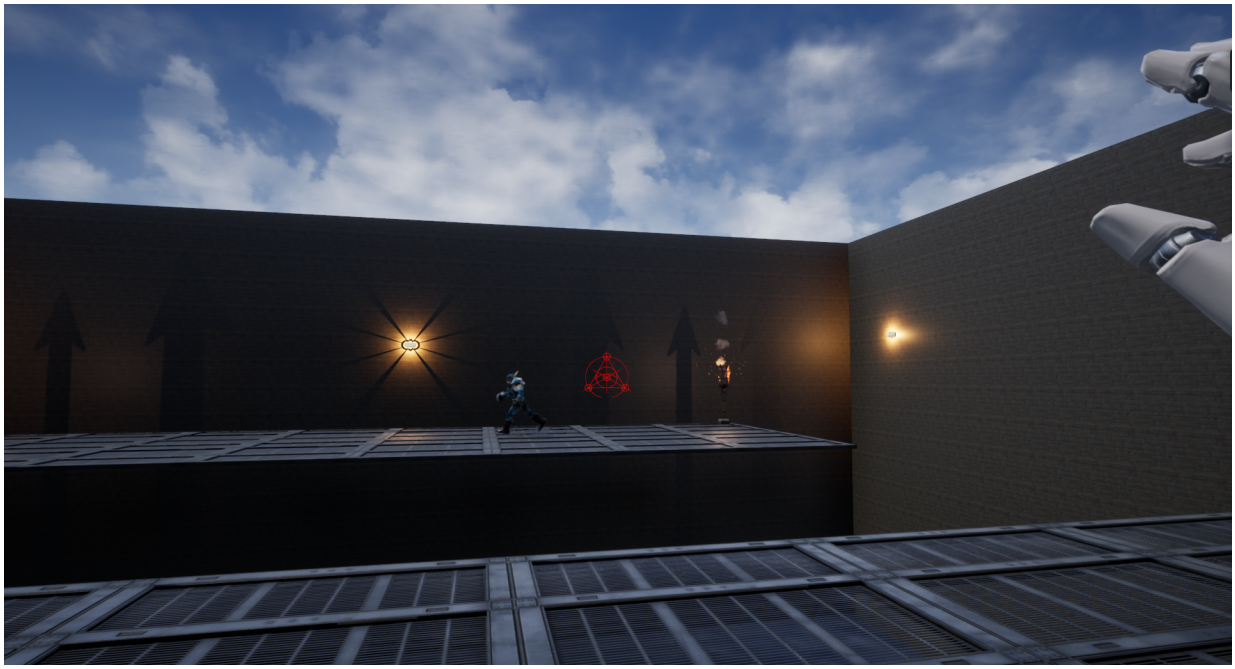


Figure 1. Catalyst screenshot

In Catalyst, the participants were tasked to shoot fireballs at a target moving back and forth across a platform. The latency delayed the mouse input and movement of the player but did not degrade the quality of the graphics. The chosen latencies for the user study were 0ms, 50ms, 100ms, and 125ms. The difficulty controlled the speed at which the target travels across the platform. The two difficulty settings had the target move at 200cm/s and 500cm/s in-game, which corresponds to what we found to be easy and difficult in our pilot studies. The length of the platform was approximately 1600cm. Finally, the attribute scaling changed the hitbox size of the target. The scale factors which were studied were 1.0, 1.5, and 2.0. For example, if the scale factor is 1.5, then the hitbox of the target is scaled in the x, y, and z-direction by a factor of 1.5.

Table 2. Independent variable values for Catalyst

Latency	0ms, 50ms, 100ms, 125ms
Difficulty	200cm/s (easy) and 500cm/s (hard)
Attribute scaling	1.0, 1.5, and 2.0

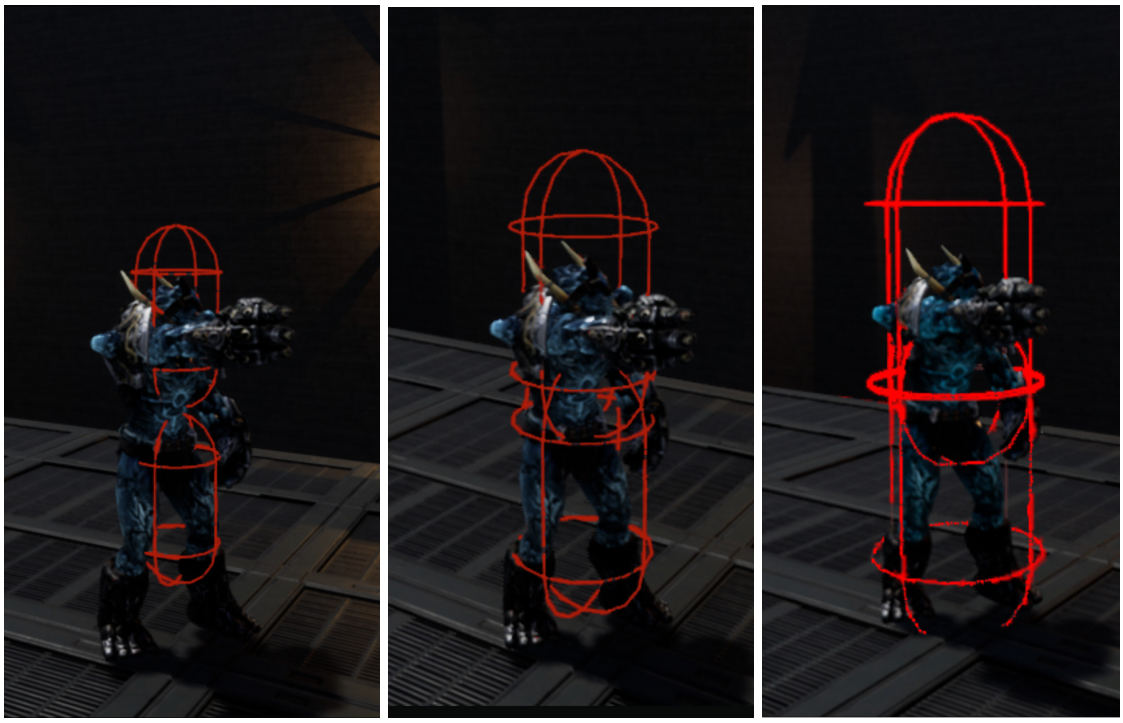


Figure 2. Catalyst hitbox size with 1.0 (left), 1.5 (middle), 2.0 (right) scalings

Section 3.1.3.2 Nova

Figure 3. Nova screenshot

In Nova, the participants were tasked to shoot cubes that spawned based on the rhythm of the music and a certain cooldown time. The chosen latencies for the user study were 0ms, 50ms, 100ms, and 125ms. The latencies delay the mouse movement and input but do not affect the graphics of the game. The difficulty, which was chosen to be 1s (easy) or 0.5s (hard), controlled the cooldown time which is the minimum time interval between note spawns. A shorter cooldown time means that the cubes spawn more frequently. Finally, the attribute scaling addressed the valid target time at which the note remained on the screen. A longer target time increases the player's opportunity to shoot the cube. The chosen attribute scalings were 1.0, 1.5, and 2.0.

Table 3. Independent variable values for Nova

Latency	0ms, 50ms, 100ms, 125ms
Difficulty	1s (easy) and 0.5s (hard)
Attribute scaling	1.0, 1.5, and 2.0

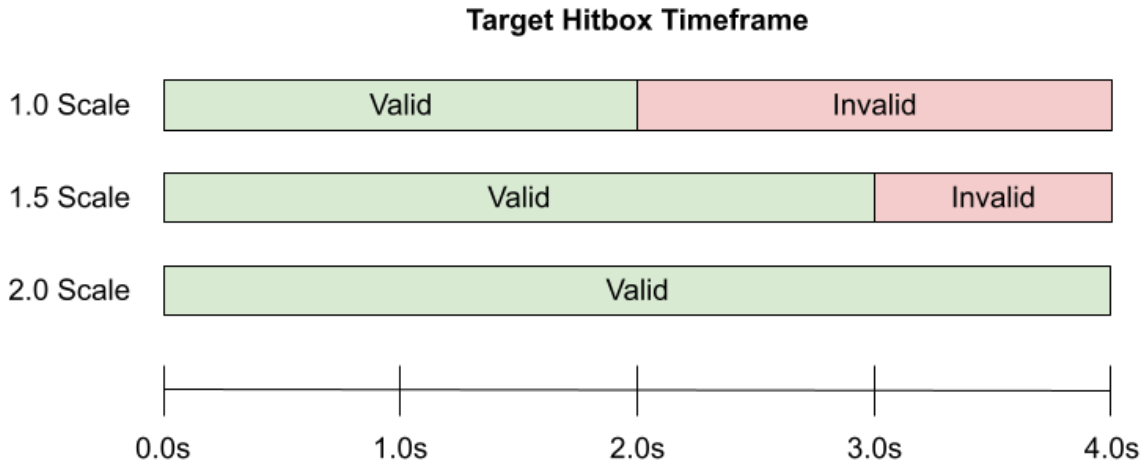


Figure 4. Nova valid target hitbox time based on attribute scaling

Section 3.2 User Study Development

This section describes the development of the user study scripts, the modifications made in the games, the composition of the quality of experience and demographic surveys, and the pilot studies.

Section 3.2.1 Script Development

Before the launch of each game, the script opened the quality of experience survey for the appropriate game in a web browser. Between each round of the game, the participant fills out this survey based on their playthrough experience from the previous round. At the end of the second game, the script opened the demographics survey.

The script generates a JSON configuration file with randomized order of settings for each game. This allows each participant to play the settings in a different order, which once again reduces the effects of the player improving their performance as they become adjusted to the setup environment. Each setting in the JSON configuration file was unique and was a combination of the three independent variables: latency, difficulty, and attribute scaling.

The user studies were launched through a Python script that randomized the gameplay order, resulting in some participants playing Catalyst first while others played Nova first. When all rounds of the first game were finished, the second game was launched. The participant might be unfamiliar with the computer settings, such as mouse sensitivity, and they could underperform as a result. In such cases, when they play the second game, their performance will improve as they have adjusted to the environment. The game randomization technique reduces the effects of game order in our general collected data.

Section 3.2.2 Game Modifications

Our team collaborated with the two MQP groups to customize their games towards conducting our user studies. The games were modified such that they meet our criteria mentioned in section 3.1.1. Each game consisted of 24 rounds with unique settings and two practice rounds. The practice round settings for Catalyst have an added latency of 0ms, a difficulty of 200 (easy), and a scale factor of 1. The practice round settings for Nova have an added latency of 0ms, a difficulty of 1s (easy), and a scale factor of 1. The settings were specified through a JSON file which is generated by the script and read in by the games. Between each round, the game pauses for a minimum of 10 seconds and prompts the user to take the quality of experience survey, and unpauses when the user is done. When the game concludes, it writes the player's actions and performance to a log file, which we saved for later analysis.

Section 3.2.3 Quality of Experience Surveys

To determine the quality of experience of the game under each condition, the participant filled out the following survey after each trial. We asked the players whether they would like to play the game again on a scale from 1 to 4 because we wanted to prevent them from always choosing a neutral opinion.

1. How enjoyable was this session? (Rate from 1-5)
2. How frustrating was the task? (Rate from 1-5)
3. How challenging was the last session? (Rate from 1-5)
4. How well do you think you performed? (Rate from 1-5)
5. How responsive were the controls? (Rate from 1-5)
6. Based on the quality of experience, how likely are you willing to play the game again? (Rate from 1-4)

Section 3.2.4 Demographics Surveys

To provide context to the research, we collected the demographics of our participants. These could reveal insights on any biases during our research and helped characterize our samples. We asked the following questions:

1. What is your name?
2. What is your age?
3. What is/are your major(s)?
4. What is your gender (Male, Female, Non-binary, Prefer to not say)?
5. How many hours on average do you spend playing games in a typical week?
6. Have you played games on a cloud-based game service before?
7. Which device do you typically play on? (Console, PC, Mobile, None)
8. How would you rate yourself as a gamer on a PC? (Rate from 1-5)
9. How would you rate yourself as a gamer on a console? (Rate from 1-5)
10. How would you rate yourself in an FPS game?

11. How would you rate yourself in a Rhythm game?
12. What kind of games do you typically play? (Check all that apply)

Section 3.2.5 Pilot Studies

The purpose of the pilot studies was to determine a reasonable duration for each round, find potential bugs in the game, and the appropriate values for latency, difficulty, and latency compensation. These pilot studies were conducted on 3 people who provided us with feedback on the design of the user studies and game bugs that they encountered.

Our objective was to limit the total duration of the user study session to 30 minutes such that the participant can remain focused on the task and their performance generates meaningful data. From the pilot studies, we determined that having 4 variations of latency, 3 variations of difficulty, and 4 variations of attribute scaling values required more than an hour of playtesting, which was too long. We reduced the number of settings to 4 variations of latency, 2 variations of difficulty, and 3 variations of attribute scaling values, such that each game has an approximate duration of 15 minutes.

The chosen latencies had to be noticeably different from each other such that they had a distinct impact on the game. However, they cannot make the game unplayable. Our testers and mentors from Google stated that latency of above 200ms became unbearable and was unrealistic, so we limited our numbers to this threshold. The selection for difficulties followed the same philosophy. The difference between the easy and hard settings should be noticeable from each other. However, the easy setting should at least provide some sort of challenge and should not invite the player to score on every shot. The hard setting provides a noticeable challenge, but the target should not be impossible to hit. Finally, the attribute scaling values were chosen such that increasing them gives an evident advantage to the player, but should not make the game incredibly easy.

The pilot studies also allowed us to improve the game or to find bugs that might become a hindrance later in the study. Throughout the pilot studies, we got the development team to change multiple factors such as cursor size, the brightness of the game, where notes could spawn, as well as other factors to improve the quality of experience of the game and to make sure that any frustration was caused by latency and not some external factor. We also found some bugs in our testing, such as notes that spawn outside of the boundaries of the game, as well as issues with how the game works on monitors with a different resolution than the ones that the MQP developers worked on.

Section 3.3 Accommodations for COVID-19

Our study was conducted amid the COVID-19 pandemic, so we took social distancing precautions recommended by the CDC. Here were the steps we took to prevent the spread of the pandemic in the lab:

1. The lab only had the research proctor and the participant scheduled for the correct allotted time inside.
2. All people inside the lab were kept 6 feet apart from each other.
3. All people inside the lab wore masks.
4. If the disease emerged on the Worcester Polytechnic Institute campus, we were prepared to shut down the study.
5. Participants used hand sanitizer before entering the lab
6. Proctors used hand sanitizer between sessions
7. The research proctor wiped down all surfaces after each trial.
8. No eating or drinking was allowed in the lab.
9. The research proctor and human participant passed their bi-weekly COVID test at WPI before entering the lab.
10. If the research proctor or human participants felt sick, then they stayed home.
11. The user study was conducted in FL A21 (the "Zoo lab")". This laboratory was cleared for re-opening, arranged by the CS Department Head (Craig Wills).

Section 3.4 Recruiting players

Before recruiting players to test the games, we had to first set up a page on the Slottr system, which is a website that allows people to sign up for certain time slots. In our case, we sent out our Slottr webpage to let people sign up for times they were available, keep a schedule on playtests, and send reminder emails to players. To get people to run tests, we had to recruit people and give incentives in exchange for their time.

Section 3.4.1 Incentives

To encourage a more general population, we had a raffle for Stadia subscription codes for people who agree to be test subjects.

Section 3.4.2 IMGD Program

One major source of players we relied on for testing was the IMGD majors at WPI, which have a requirement to do general playtesting. These players are likely to be more experienced at games, and so to get less experienced players, we also recruited from other places.

Section 3.4.3 Psychology Program

The psychology program of WPI has an online participation tool (SONA) for participants to sign up for the study and receive credit in psychology courses for participating. Our team used this tool to gather participants for our study who may not be as experienced in video games as the IMGD majors. To use the participation tool, we needed to submit a statement to the psychology department that includes:

1. All researchers involved
2. Experiment description
3. Experiment duration
4. IRB approval form, consent form

Section 3.4.4 Institutional Review Board (IRB)

To test on human participants, our study needed to be reviewed by WPI by submitting an IRB application to make sure that the mental, physical, and financial safety of the participants will not be violated. We needed to make sure all the participants were over 18 or older and reconfirm this in the IRB application as well. Our IRB application contains a brief description of our experiments, the method listing, potential risks, the methodology chapter section, and our sample survey questions.

Section 3.4.5 Trouble Recruiting

After a user signs up for a test session on the Slottr page, the player is reminded the day before the day of the test. However, even with the reminder email some players still missed some testing sessions due to a variety of factors. One common problem was the location of the lab, as players sometimes could not find their way there due to the confusing layout of the building and naming of the floors. Sometimes users would also not show up to testing sessions without notifying us, causing us to lose a valuable data point. Another cause of trouble in the recruiting process is that the coronavirus caused fewer people to want to sign up for a playtesting session, since even though the lab is clean and we follow all protocols, some users may still be nervous. This caused us to have fewer participants than we might have in a normal year.

Section 3.5 User Procedure

Each participant underwent the same procedure during the study. When the user studies were conducted, the participant entered the lab and followed the procedures described in this section. They performed the study at a computer that has the Python script and both games installed.

Section 3.5.1 Hardware Specifications

The table below shows the hardware which we used to conduct our user studies. Furthermore, we measured the local latency which was 80ms. We measured this using a high-speed camera (1000 frames/second) to record a shooting action. Then, we used Openshot, a video analysis software to observe the footage frame-by-frame, where one frame is equivalent to 1ms passing. The local latency was calculated by subtracting the time when we completely pressed down the left-mouse button and the time when the spark of the gun fire appears on the screen.

Table 4. User study hardware specifications

CPU	Intel i7-4790K at 4.00GHz x 8 cores
RAM	16 GB
Graphics Card	Intel HD Graphics 4600, NVIDIA GeForce GTX 960
Mouse	Logitech G203 8k DPI, 1000 Hz
Monitor	Dell U2412M, 1920x1200, 24”, 16:10, 60Hz
OS Version	Windows 10

Section 3.5.2 User Study Procedure Overview

During the user study, the participant adhered to the following procedure:

1. Start the script
2. Play the first game
3. Play the second game
4. Fill out the demographics survey

For each round of a game, the procedures below were followed:

1. Start or resume the game to begin the next level
2. The participant was given 20 seconds to play the game.
3. Their performance was recorded by the game into a log file
4. The participant fills out quality of experience survey

Section 3.5.3 Consent forms

Before conducting the tests, we were supposed to have the users sign off on a consent form that we had written. However, due to a communication error, we did not have the users sign off on the forms at the time of the study, and we sent the consent forms through email at a later date.

Section 3.6 Data Collection and Analysis

After the end of each trial, the game recorded the player’s accuracy among other factors to a log file for later parsing. The participant also filled out a survey asking about their experience playing the game after each trial. These metrics help us determine how increasing latency and attribute scaling affect the player’s performance and quality of experience. After the experiment ends, the participants fill out a demographic survey which asks for their general information, abilities in gaming, and experience with cloud-based gaming.

We ran statistical analysis and made visual diagrams based on the data we had collected. The first step in this process was to standardize the data that comes in from the user experience surveys, the demographics survey, and the performance logs from both of the games. One of the major roadblocks here was that even after many pilot tests, there were still bugs that arose from unexpected circumstances, such as a user having to leave early which can corrupt or cause a datapoint to become unusable. Users also frequently entered inconsistent names across different forms, causing the data to have to be cleaned manually before the scripts can properly process the data. From here, we made various scripts in the Python scripting language to create various visualizations and analyses of the data to better understand the relationship between the various factors in the experiment.

Section 4 Results and Analysis

This chapter presents the results and analysis of the collected demographics, performance, and quality of experience data from the user studies.

Section 4.1 Demographics

A participant left their session early without finishing the entire user study. Exiting the game midway caused an error in the system for one participant which made the log files unreadable and impossible to parse. The data points collected from this participant were manually removed because they were unusable.

Since our recruitment source was a polytechnic institute, most participants were young, male students. More specifically, the average age of our participants was 19.8 years with a standard deviation of 1.5 years. Among our 23 participants, 15 were male, 6 were female, 1 was non-binary, and 1 decided to not disclose their gender. They played on average 10.4 hours of video games every week with a standard deviation of 8.44 hours. The table below presents a breakdown of their majors at our institution.

Table 5. The major breakdown of the user study participants

Major	Percentage
Computer Science	30%
Interactive Media & Game Development	17%
Computer Science and Interactive Media & Game Development	13%
Physics	13%
Others	26%

Section 4.2 Data Sorting and Outlier Removal

To extract useful results and data from our performance logs and surveys, we first unified them into a single format to allow for easy manipulation. This was achieved through parsing the log files, produced from the user studies, line by line using a Python script. Then, we combined them with the corresponding user survey data to make a single CSV containing all information for a particular game. An obstacle that we encountered was that some users did not refer to themselves by the same name between the surveys and performance logs, forcing us to manually parse through some data to correct these errors.

Using this organized data, we explored trends and outliers. For some rounds, we noticed that the user shot an unusually high or low number of times. We assumed that, for these rounds, the user was either not focused on accuracy and was shooting as much as possible, or was not trying. Since these rounds were also associated with abnormally high or low accuracies, we removed all rounds in which the number of times the user shot was more than

$1.5x interquartile\ range$ above the third quartile or lower than the first quartile. This removed 7.4% of the data points for Nova and 2.4% of the data points for Catalyst.

Section 4.3 Performance

For our study, we measured the performance of the player based on accuracy. For Catalyst, we define accuracy as the number of successful hits over the number of total shots. For Nova, accuracy is the number of successful hits over the total number of targets that were spawned during a round. Before the experiment, we hypothesized that as latency and difficulty rise, accuracy will decrease because both of these factors inhibit user performance and make it harder for users to achieve their intended outcomes. We also hypothesized that as the attribute scale for the games increase, the accuracy will increase as well. This is because, for Catalyst, a larger scale means that there is less spatial precision required from the user. For Nova, the scale increases the duration at which the note remains on the screen, allowing the player to have more time to react and destroy notes.

Section 4.3.1 Accuracy vs. Latency

Figure 5 showcases the relationship between added latency (in ms) and accuracy (in percent) through boxplots. The independent variable is added latency in milliseconds, which is the artificial delay injected into the games, not considering local latency. The dependent variable is percent accuracy. The colored circles depict the means of the data. The horizontal bar in the rectangular shape illustrates the median. The lower edge of the rectangle is the first quartile and the higher edge is the third quartile as they define the interquartile range which contains the middle 50% of the data. The whiskers are the extremities of the boxplot and are defined as $quartile \pm 1.5\ x interquartile\ range$. The unfilled points are the outliers of the boxplot. For both games, as latency increases, the accuracy tends to decrease.

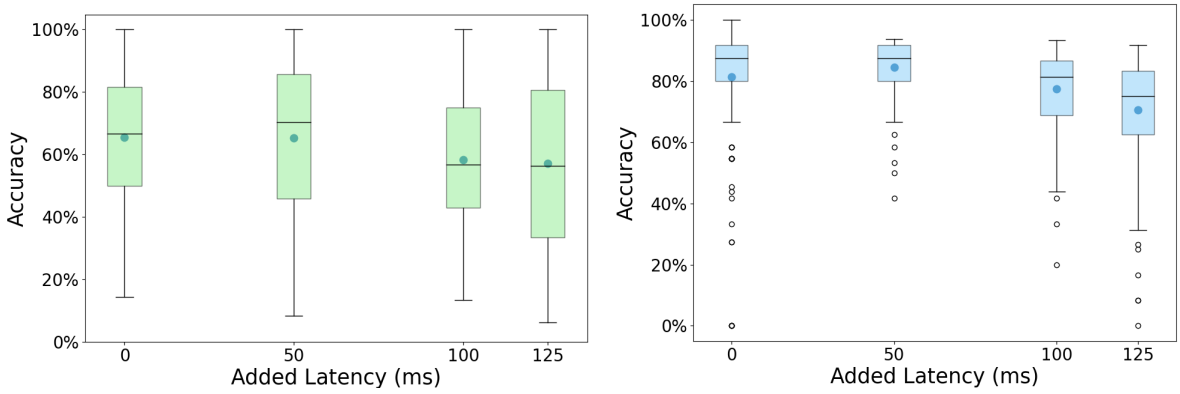


Figure 5: Accuracy vs. latency for both games, Catalyst (left) and Nova (right)

Section 4.3.2 Accuracy vs. Difficulty

Figure 6 demonstrates the relationship between difficulty (easy or hard), the independent variable, and the accuracy in percentage, the dependent variable, through boxplots. For both games, as the difficulty increases, the accuracy decreases.

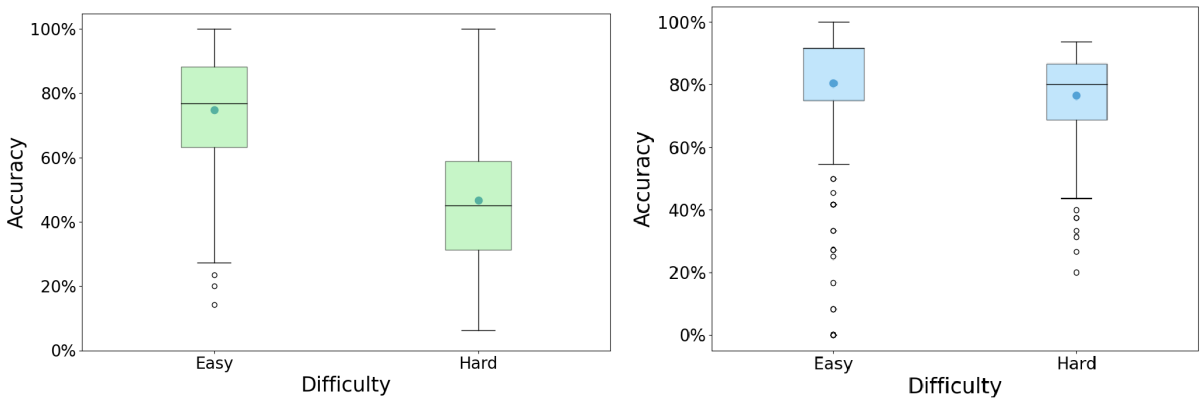


Figure 6: Accuracy vs. difficulty graphs for both games, Catalyst (left) and Nova (right)

Section 4.3.3 Accuracy vs. Scale

Figure 7 demonstrates the relationship between scale, the independent variable, and the accuracy in percentage, the dependent variable, through boxplots. For both games, as the scale increases, the accuracy also increases. This supports the idea that attribute scaling could help reduce the effects of latency as the loss in performance due to latency could help balance out with the increase in performance due to the attribute scaling.

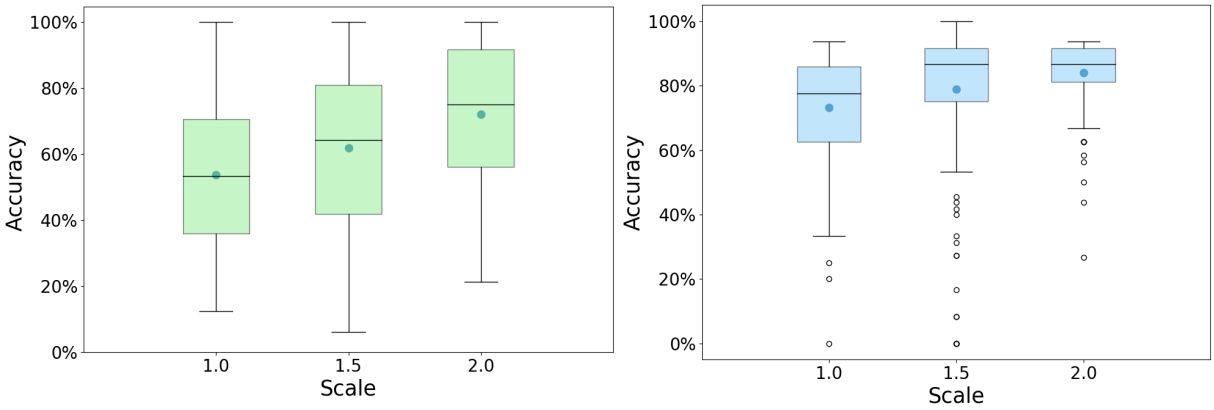


Figure 7: Accuracy vs. scale graphs for both games, Catalyst (left) and Nova (right)

Section 4.3.4 Compensation Models

One of the main ways we modeled the data was through making a linear regression between the independent variables of latency, scale, and difficulty, and the dependent variable, accuracy. After taking the average accuracy for each combination of values, we took a linear regression for mean values with a Python script.

For Nova, the formula is:

$$accuracy = -0.001 \cdot latency + 0.11 \cdot scale + 0.1 \cdot difficulty + 0.62$$

with accuracy being measured as a fraction, and latency is measured in milliseconds. The corresponding formula for Catalyst is:

$$accuracy = -0.0008 \cdot latency + 0.2 \cdot scale + 0.001 \cdot difficulty + 0.72$$

These models resulted in an R^2 value of 0.67 for Nova, and an R^2 of 0.95 for Catalyst.

An example of using the model for compensating for latency with each game is shown below. The x-axis is the added latency in milliseconds, and the y-axis is percentage accuracy. The blue line represents how the accuracy varies as the latency varies when the scale is set to one. The orange line represents an ideal accuracy that does not vary with latency and each point on the orange line shows the scale needed at a particular latency. These scale values are 0.85, 1.05, 1.25, and 1.34 for Catalyst, and 0.66, 1.14, 1.62, 1.85 for Nova.

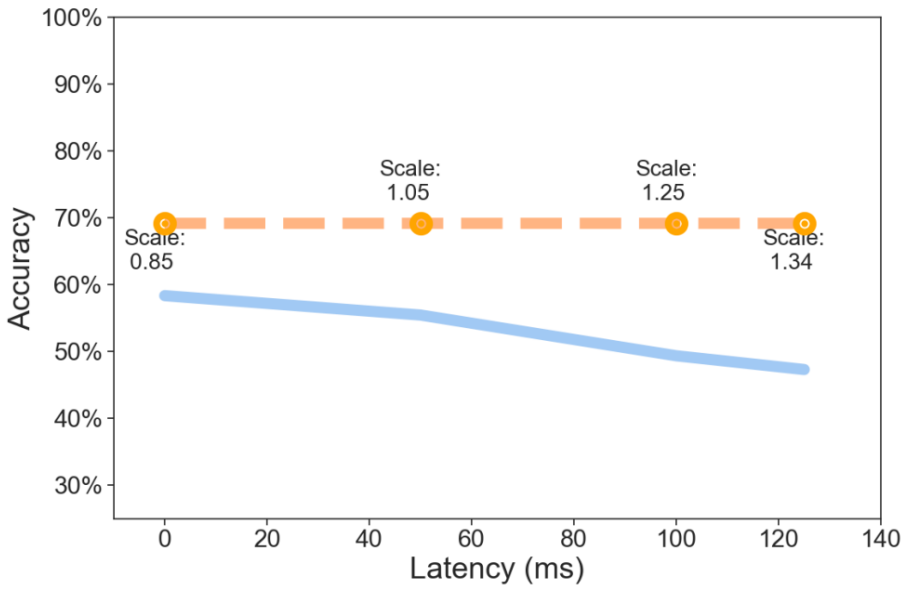


Figure 8: Comparison between the average accuracy at different latencies for Catalyst in blue and the ideal accuracy when compensating by changing the scale in orange

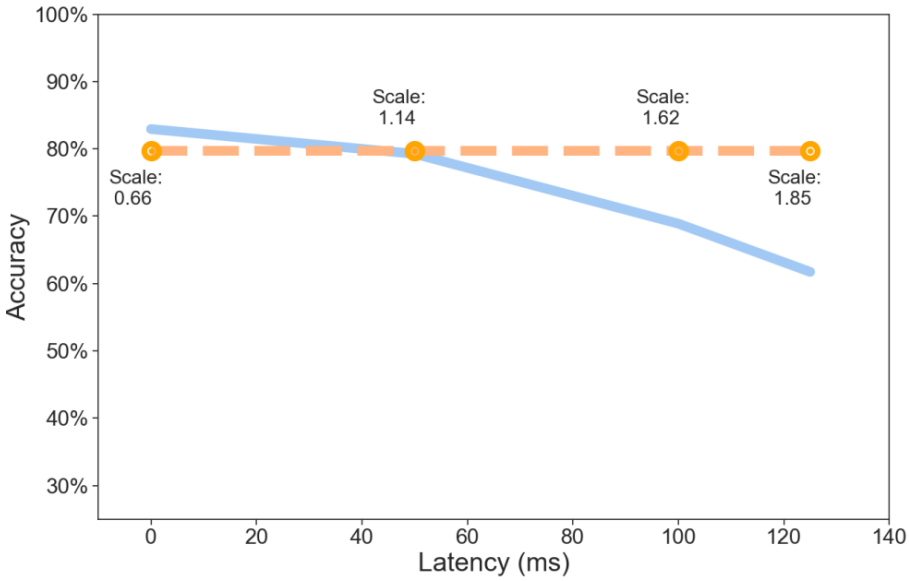


Figure 9: Comparison between the average accuracy at different latencies for Nova in blue and the ideal accuracy when compensating by changing the scale in orange

Section 4.3.5 ANOVA Test for Performance

We conducted an ANOVA test to see if changes in our independent variables are related to significant differences in our accuracy.

Table 6. ANOVA test for Catalyst game

Independent Variable	P Value	F-Value
Latency	< .001	5.2
Scale	< .001	33.1
Speed (Difficulty)	< .001	6.9

Table 7. ANOVA test for Nova game

Independent Variable	P Value	F-Value
Latency	< .001	16.7
Scale	< .001	16.9
Cooldown (Difficulty)	< .001	7.1

From Table 6 and Table 7, all P values are less than 0.01. At a significance level of less than 0.01, we can reject the null hypothesis which states that there is no significant effect on the accuracy by the independent variables of latency, scale, and difficulty. This, combined with our trends shown above, show that latency and difficulty have a significant negative effect on user

performance for both games, while attribute scaling has a significant positive effect on user performance.

Section 4.3.6 Cohen's D Test for Performance

We also performed Cohen's D tests to see the effect that particular changes in our independent variables had on the accuracy. Below you can see the comparisons of different variables against the baseline values of 0ms latency, scale of 1, or difficulty of easy.

Table 8. Cohen's D test results for Catalyst performance data

Cohen's D Variable	Cohen's D Effect Size
Latency | 0ms vs 50ms	-0.20
Latency | 0ms vs 100ms	0.32
Latency | 0ms vs 125ms	0.36
Scale | 1 vs 1.5	-0.35
Scale | 1 vs 2	-0.79
Difficulty | Easy vs Hard	1.48

Table 9. Cohen's D test results for Nova performance data

Cohen's d variable	Cohen's d effect size
Latency | 0ms vs 50ms	-0.20
Latency | 0ms vs 100ms	0.22
Latency | 0ms vs 125ms	0.58
Scale | 1 vs 1.5	-0.31
Scale | 1 vs 2	-0.75
Difficulty | Easy vs Hard	0.26

Above, you can see that as the values grew farther from their baseline values, the effect size tended to grow too. This indicates that a larger difference from the baseline values makes a bigger effect on user performance.

Section 4.4 Quality of Experience

Our quality of experience survey collected a subjective measure of the participant's opinion on each game configuration. The ANOVA test results are shown in Table 10. All P-values were less than 0.01. At a significant level of less than 0.01, we can reject the null

hypothesis which states that latency does not affect enjoyability, frustration, perceived difficulty, perceived performance, perceived responsiveness, and replay likelihood for both games. Therefore, we were able to determine that latency had a significant effect on these quality of experience factors for both games.

Table 10. ANOVA test for the quality of experience data (refer to section 3.2.3 for full QoE questions)

	Catalyst: P Value	Catalyst: F-Value	Nova: P Value	Nova: F-Value
Enjoyability	< 0.001	6.94	< 0.001	16.7
Frustration	< 0.001	5.9	< 0.001	22.1
Perceived Difficulty	0.009	3.9	< 0.001	14.5
Perceived Performance	< 0.001	5.8	< 0.001	24.1
Perceived Responsiveness	< 0.001	11.7	< 0.001	23.6
Replay Likelihood	< 0.001	7.0	< 0.001	14.9

With this information, we can verify that latency does have a significant effect on the quality of experience of our games, however, it does not verify whether the effect of latency is a positive or a negative one for each factor. Below are graphs showing the relation between latency and each survey question from the user studies. The circle for each latency shows the average response from the survey which was measured on a scale from 1 to 5. The bars show the 95% confidence interval for that particular latency.

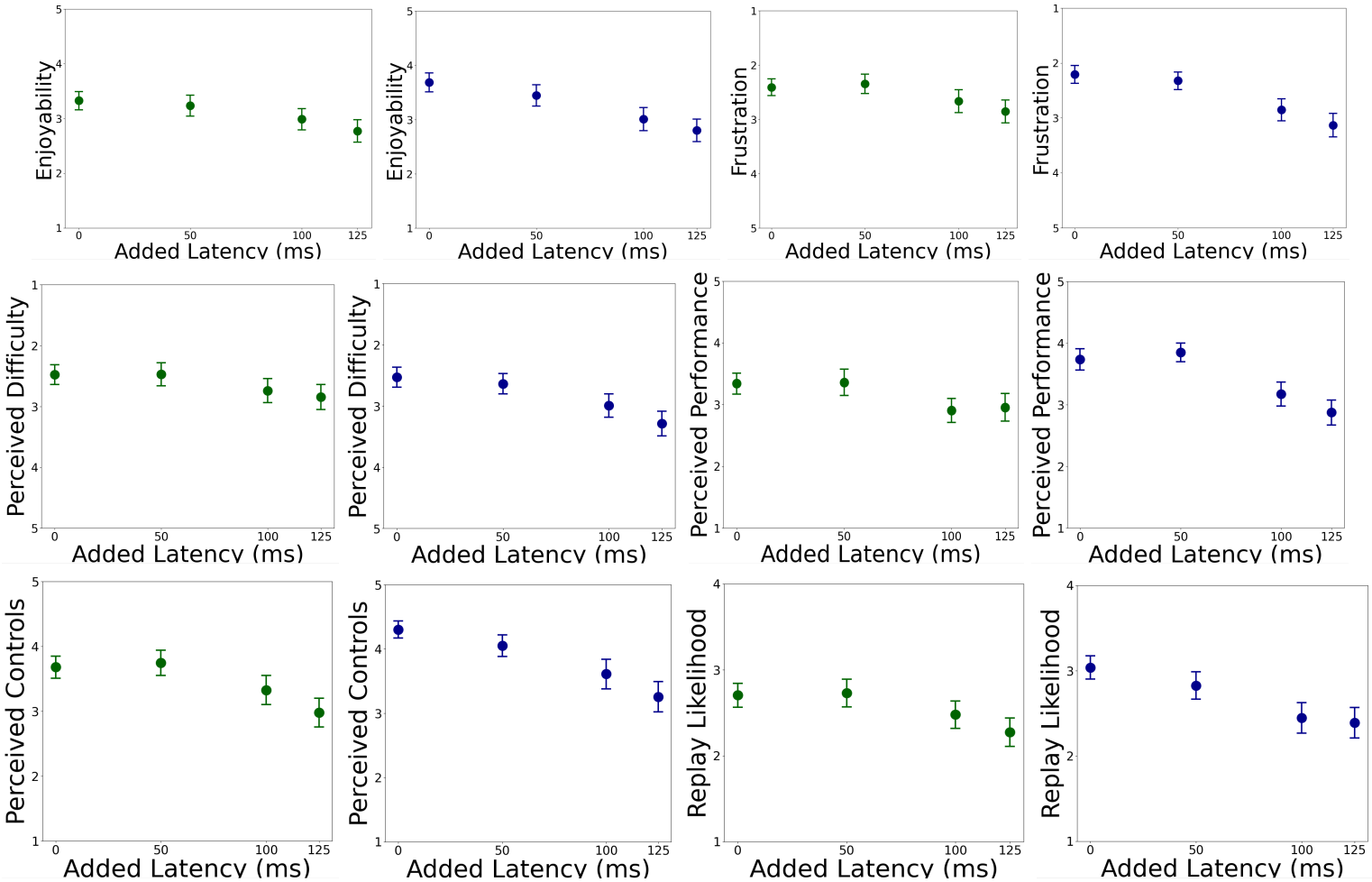


Figure 10-21. QoE Response vs. Added Latency (Refer to Section 3.2.3 for full questions)

Figures 10-21 demonstrate the relationship between quality of experience responses from the surveys and the added latency in milliseconds. As the latency increases, players have indicated that their enjoyment decreases as their frustration increases. Furthermore, increased latency makes the game seem more difficult and less responsive. As latency increases, the responses show that players are less likely to replay the game.

We conducted another Cohen's D test to see the effect that changing latency, scaling, and difficulty had on user enjoyment.

Table 11. Cohen's D test results for Catalyst quality of experience data

Cohen's d variable	Cohen's d effect size
Latency | 0ms vs 50ms	0.06
Latency | 0ms vs 100ms	0.30
Latency | 0ms vs 125ms	0.46
Scale | 1 vs 1.5	0.10

Scale | 1 vs 2	0.31
Difficulty | Easy vs Hard	0.16

Table 12. Cohen's D test results for Nova quality of experience data

Cohen's d variable	Cohen's d effect size
Latency | 0ms vs 50ms	-0.20
Latency | 0ms vs 100ms	0.57
Latency | 0ms vs 125ms	0.80
Scale | 1 vs 1.5	-0.20
Scale | 1 vs 2	-0.19
Difficulty | Easy vs Hard	0.14

Above you can see that as the values, excluding the scale for Nova, grew farther from their baseline values the effect size tended to grow too. This indicates that a larger difference from the baseline value makes a bigger effect on user performance.

Section 5. Future Work

While our methodology was effective, we have identified some areas of improvement which could advise future research groups focusing on similar topics. This section proposes opportunities for future studies which are extensions of our work.

The selected games for the user studies were developed by two MQP groups at our institution. During the pilot study phase, we noticed some bugs which could have been detrimental to the data collection. For example, a target occasionally spawned at the end of a Nova round. This created a scenario where the participant couldn't shoot the target, which decreased the recorded accuracy and corrupted the data. An interesting direction of research would be to reproduce our studies using a popular title with more complex mechanics. Established games are generally more polished, reducing the chances of encountering these difficulties and increases in the chances of obtaining more reliable results. Additionally, our games implemented simple mechanics, whereas commercial games typically have multiple or complex mechanisms.

For our user studies, we were only able to test four added latency values, two difficulty settings, and two attribute scalings. This was due to user time constraints because we wanted to prevent our user session from exceeding a thirty-minute limit. Future studies could focus on only one independent variable, allowing them to explore a wider range of values at smaller increments. This would increase the domain of collected data points, perhaps resulting in a more accurate and robust compensation model.

We chose 125ms as our maximum added latency because we believed that 150ms would have made the game unplayable. However, with further testing, it would have been bearable and would have made our latency increments more consistent for data visualization. Additionally, our added latencies were constant throughout each round. Another future study would be to introduce jitter (varying latency) during the round. This may better simulate some gaming experiences.

The two MQP groups will implement our compensation models in their games to validate them and determine their effectiveness. Future research teams could apply these models in other games to test their versatility.

Finally, we propose that our research be re-created with games that implement different actions with different attributes to scale. Our study focused on first-person shooters, rhythm games, and attribute scaling, while there are other genres worth studying.

Section 6. Conclusion

Cloud games have emerged in recent years with the potential to revolutionize the gaming industry. They alleviate the processing burden from local computers and make gaming more accessible across different platforms. However, cloud games are susceptible to latency because user input must travel from the local computer to data centers for processing, and then a video stream is returned to update the current game state. This added latency could be detrimental to player performance and their satisfaction with the game, leading to anger and frustration. As a result, latency compensation methods were introduced to address this issue.

Our research aimed to determine how latency affects player performance and quality of experience, and how games could use attribute scaling to compensate for added latency. We performed a user study with twenty-three participants where users played two games titled Nova and Catalyst in random order with added latency. Catalyst is a first-person shooter game that scales the target hitbox size based on added latency. Nova is a rhythm game that scales the duration at which the target remains on the screen based on added latency.

Participants played 24 rounds for both games, each with a different combination of the independent variables: latency, scale, and difficulty. The added latencies were 0ms, 50ms, 100ms, and 125ms. The two difficulties were determined through pilot study testing and were labeled “easy” and “hard”. The attribute scaling factors were 1.0, 1.5, 2.0. The participants filled out a quality of experience survey after each round, providing a qualitative assessment. Furthermore, user performance was recorded at the end of each session for the quantitative data.

Our analysis shows that increased latency significantly decreases player performance and enjoyability, suggesting that latency in cloud games should remain a concern. Increased difficulty significantly decreases player performance. Our results also show that increased attribute scaling significantly improves player performance, hinting that attribute scaling may be an effective method at compensating for latency. Game developers and thin client service developers should be aware of the added latencies from the cloud and consider implementing compensation techniques, such as attribute scaling, to improve player performance and enjoyment.

Section 7. Acknowledgments

We would like to acknowledge the following people for making our research possible.

Primary Advisor
Mark Claypool <claypool@wpi.edu>

Thank you to our advisor, Mark Claypool, for providing us with this opportunity, guiding us through the research process, giving us feedback, and proofreading our paper.

Secondary Advisor and Lab Proctor
Xiao Kun Xu <xxu11@wpi.edu>

Thank you to our proctor, Xiao Kun Xu, for providing us with valuable advice and successfully running user studies amidst a global pandemic (while respecting COVID guidelines).

MQP Team 1 (Catalyst)
Alejandra Garza <agarza@wpi.edu>
Joseph Swetz <jsswetz@wpi.edu>
Cameroon Person <caperson@wpi.edu>
Adam Desveaux <acdesveaux@wpi.edu>
James Plante <jplante@wpi.edu>

Thank you to the Catalyst team for developing a game tailored to our user studies.

MQP Team 2 (Nova)
Michael Bosik <mbosik@wpi.edu>
Nina Taurich <ntaurich@wpi.edu>
Alex Hunt <amhunt@wpi.edu>

Thank you to the Nova team for developing a game tailored to our user studies.

Section 8. Appendix

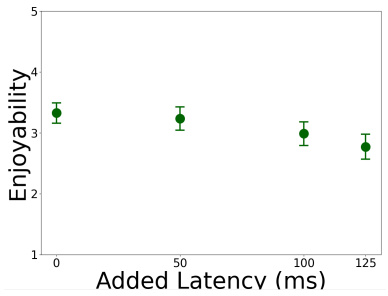
Section 8.1 Demographics Survey

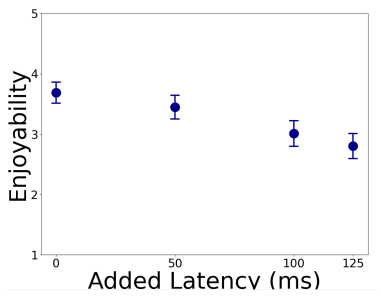
- Name
- Age
- Major(s)
- Gender
- How many hours on average do you spend playing games in a typical week?
- Have you played games on a cloud-based game service before?
- Which device do you typically play on
- How would you rate yourself as a gamer on a PC? (1-5)
- How would you rate yourself as a gamer on a console (1-5)
- How would you rate yourself in a FPS game? (1-5)
- How would you rate yourself in a Rhythm game? (1-5)
- What kind of games do you typically play? (Check all that apply)
    - First/Third person shooter (CS:GO, Halo, Fortnite)
    - MOBA (Dota 2, League of Legends)
    - Action Game (Genshin Impact, Devil May Cry)
    - Turn based RPG (Pokemon, Dragon Quest)
    - Arcade/bullet hell (Touhou, Enter the Gungeon)
    - Side scroller (Hollow Knight, Mario)
    - 3D Platformer (Mario 64, Mario Galaxy)
    - Strategy Game (Endless Space, Starsector, Civ 5, 100% Orange Juice)
    - Puzzle (Professor Layton, Puyo Puyo Tetris)
    - VR (Beatsaber, Half life Alyx)
    - Racing (Forza, Mario Kart)
    - Flight/space simulator (Flight Simulator, Elite Dangerous)

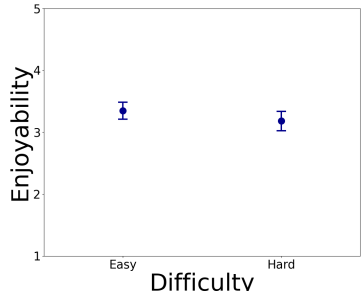
Section 8.2 Quality of Experience Survey

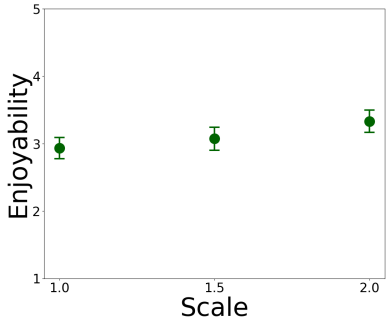
- How enjoyable was this session? (1-5)
- How frustrating was the task? (1-5)
- How challenging was the last session? (1-5)
- How well do you think you performed? (1-5)
- How responsive were the controls? (1-5)
- Based on the quality of experience, how likely are you willing to play the game again? (1-4)

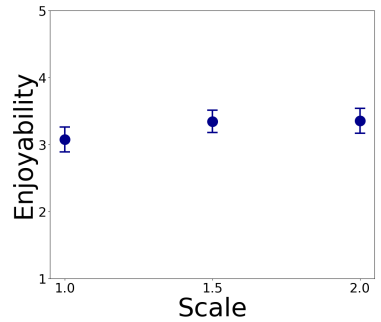
Section 8.3 Quality of Experience Graphs

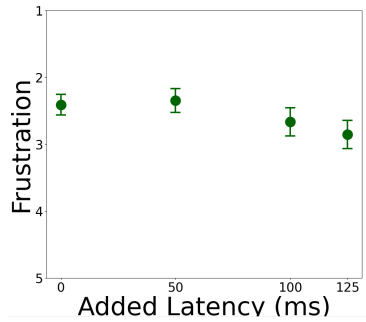


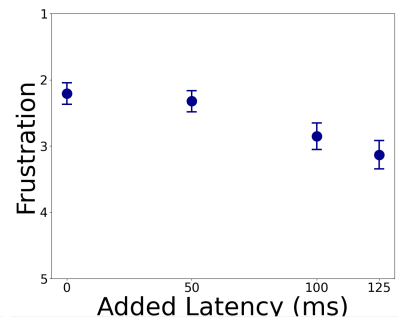


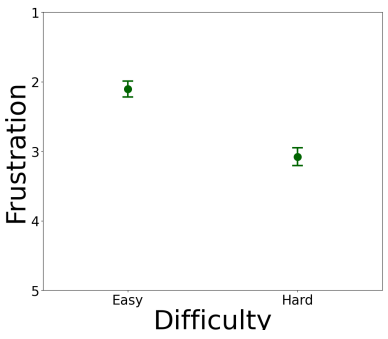


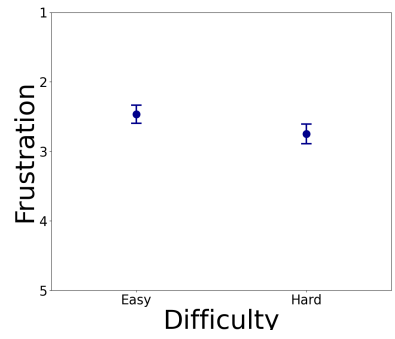


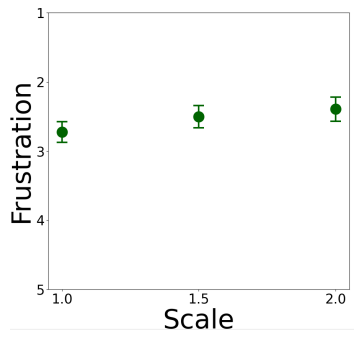


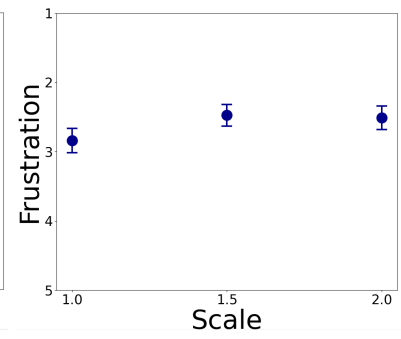


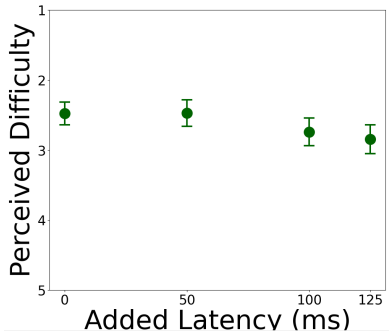


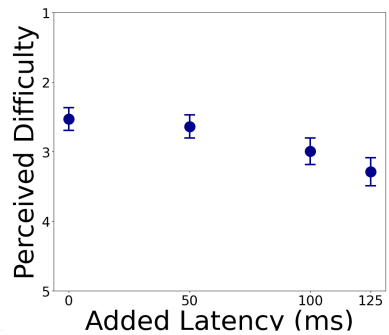


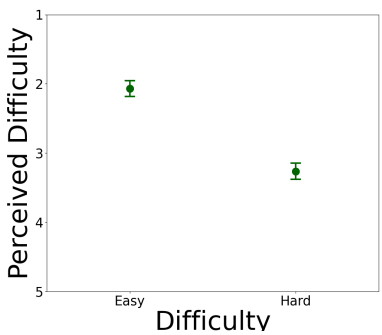


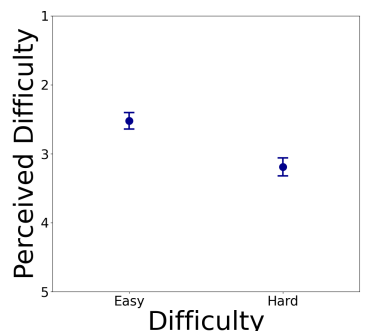


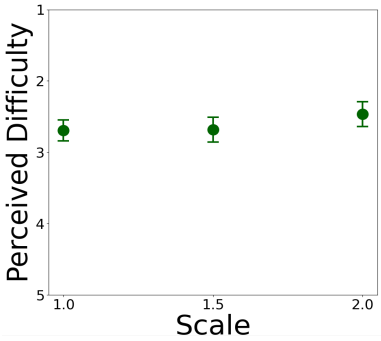


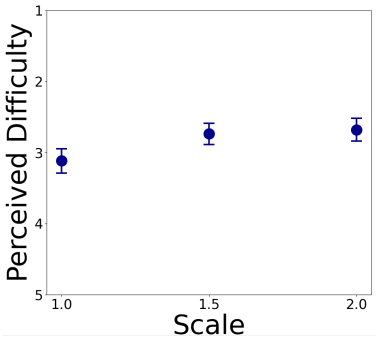


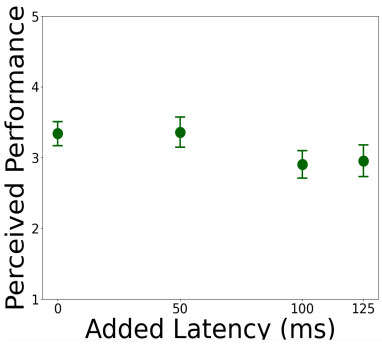


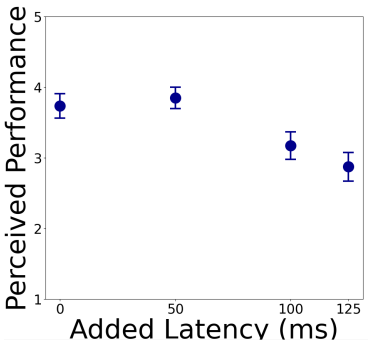


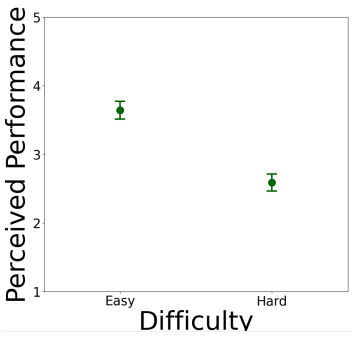


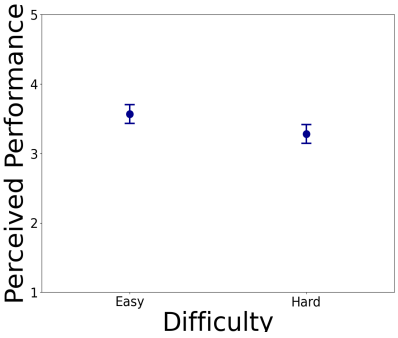

Perceived Difficulty
1
2
3
4
5
1.0
1.5
2.0
Scale

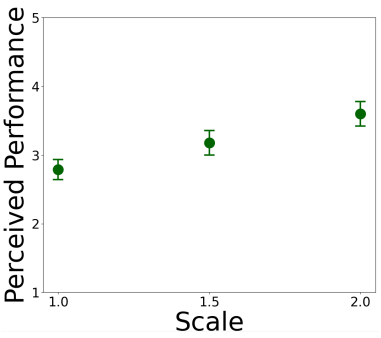

Perceived Difficulty
1
2
3
4
5
1.0
1.5
2.0
Scale

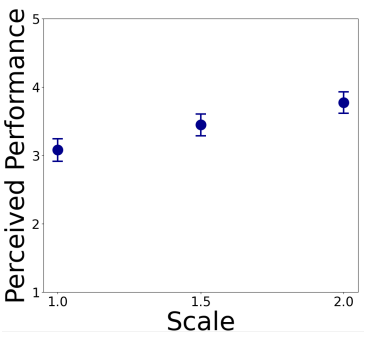

Perceived Performance
1
2
3
4
5
0
50
100
125
Added Latency (ms)

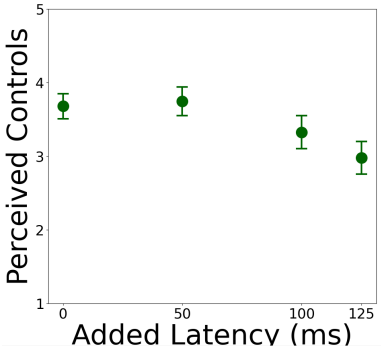

Perceived Performance
1
2
3
4
5
0
50
100
125
Added Latency (ms)

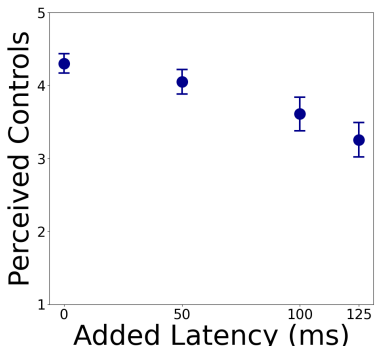

Perceived Performance
1
2
3
4
5
Easy
Hard
Difficulty

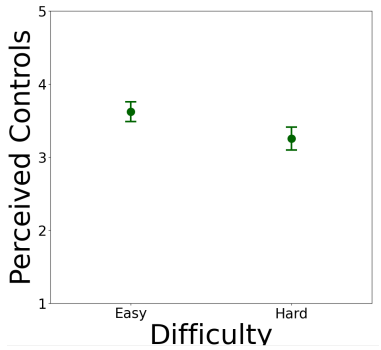

Perceived Performance
1
2
3
4
5
Easy
Hard
Difficulty

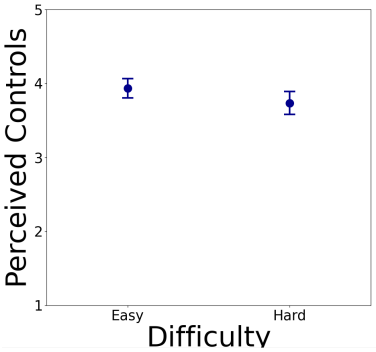

Perceived Performance
1
2
3
4
5
1.0
1.5
2.0
Scale

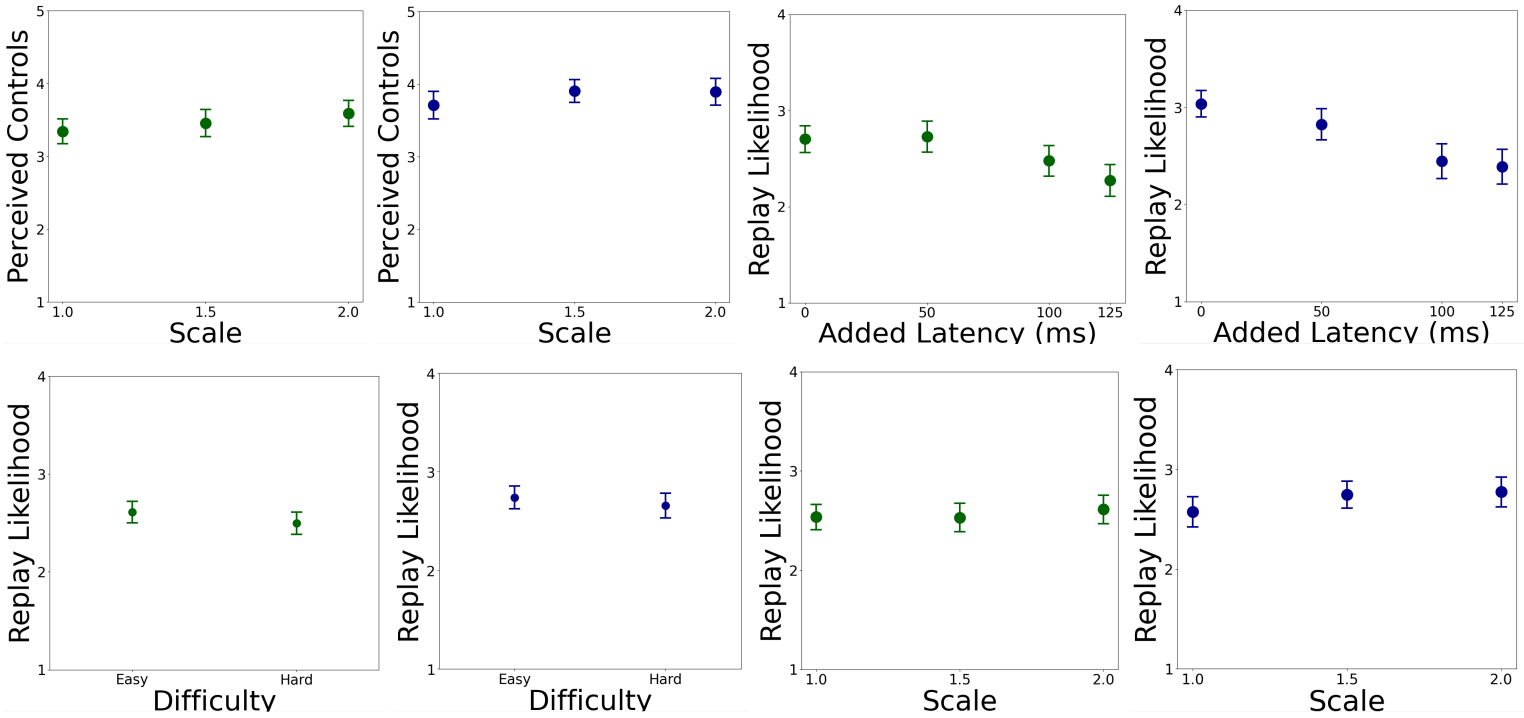

Perceived Performance
1
2
3
4
5
1.0
1.5
2.0
Scale


Perceived Controls
1
2
3
4
5
0
50
100
125
Added Latency (ms)


Perceived Controls
1
2
3
4
5
0
50
100
125
Added Latency (ms)


Perceived Controls
1
2
3
4
5
Easy
Hard
Difficulty


Perceived Controls
1
2
3
4
5
Easy
Hard
Difficulty



Section 8.4 User Study Script

```
import shutil
import os
import subprocess
import time

CATALYST_DOCUMENTS = "~\\Documents\\catalyst"

if not os.path.exists(os.path.expanduser(CATALYST_DOCUMENTS)):
  os.makedirs(os.path.expanduser(CATALYST_DOCUMENTS))

for filename in os.listdir(os.path.expanduser("~\\Documents\\catalyst")):
  if filename.startswith("log_"):
    os.remove(os.path.expanduser("~\\Documents\\catalyst\\") + filename)

for filename in os.listdir(os.path.expanduser("Games\\NovaTestBuild\\StadiaMQP")):
  if filename.startswith("output"):
    os.remove("Games\\NovaTestBuild\\StadiaMQP\\" + filename)

exec(open('ConfigNova.py').read())
exec(open('ConfigCatalyst.py').read())

shutil.move("trial_data.json", os.path.expanduser("~\\Documents\\catalyst\\trial_data.json"))
shutil.move("inputs.txt", "Games\\NovaTestBuild\\StadiaMQP\\inputs.txt")

p = subprocess.Popen("Games\\WindowsNoEditor\\CatalystDemo.exe")
```

```
while p.poll() is None:
  continue

p = subprocess.Popen("Games\\NovaTestBuild\\StadiaMQP.exe")

while p.poll() is None:
  continue

# Catalyst: move results to Results folder
for filename in os.listdir(os.path.expanduser("~\\Documents\\catalyst")):
  if filename.startswith("log_"):
     noExtension = os.path.splitext(filename)[0]
     shutil.move(os.path.expanduser("~\\Documents\\catalyst\\" + filename), "Results\\Catalyst\\" + filename)

# Nove: move results to Results folder
for filename in os.listdir("Games\\NovaTestBuild\\StadiaMQP"):
  if filename.startswith("output"):
     noExtension = os.path.splitext(filename)
     shutil.move("Games\\NovaTestBuild\\StadiaMQP\\" + filename, "Results\\Nova\\" + filename +
time.strftime("_%Y.%m.%d-%H.%M.%S"))
```

Section 8.5 Catalyst Script

```
import json
import random

delays = [0, 0.025, 0.050, 0.100] #[0, 0.025, 0.050, 0.100]
difficulties = [200, 400]
scaleFactor = [1, 1.5, 2]
allTrials = []

trainRound = {
                    "scaleFactor": 1,
                    "difficulty": 200,
                    "delay": 0
                }

for x in delays:
    for y in difficulties:
        for z in scaleFactor:
            allTrials.append({
                    "scaleFactor": z,
                    "difficulty": y,
                    "delay": x
                })

print(len(allTrials))
```

```
random.shuffle(allTrials)
trialCopy = allTrials.copy()
trialCopy.insert(0,trainRound)
trialCopy.insert(0,trainRound)
print(len(trialCopy))
toDump = {'allTrials' : trialCopy,
"time":  20.0,
"cooldown": 0.5}
with open(f"trial_data.json", 'w') as json_file:
    json.dump(toDump, json_file)
```

Section 8.6 Catalyst Script

```
import json
import random

delays = [25.0, 50.0, 100.0, 125.0]
difficulties = [0.5, 0.1]
factors = [0.5, 1.0, 2]

allTrials = []
testDuration = 20;

for x in delays:
    for y in difficulties:
            for z in factors:
                    allTrials.append([x, testDuration, y, x/z])

random.shuffle(allTrials)
toDump = {'allTrials' : allTrials}

with open(f"inputs.txt", 'w') as f:
    f.write(f"0,20,0.5,100|\n")
    f.write(f"0,20,0.1,100|\n")
    f.write(f"0,20,0.5,100|\n")  # Easy
    f.write(f"0,20,0.1,100|\n")  # Hard
    for i in allTrials:
            f.write(f"{i[0]},{i[1]},{i[2]},{i[3]}|\n")
```

Section 9. Documents

Section 9.1 Consent Forms

Informed Consent Agreement for Participation in a Research Study

Investigator: Edward Kazuya Carlson Email: ekcarlson@wpi.edu,
Tian Fan Email: tfan@wpi.edu
Zijian Guan Email: zguan@wpi.edu

Advisor: Mark Claypool Email: claypool@wpi.edu

Contact Information:
IRB Manager: Ruth McKeogh, Tel. 508 831- 6699, Email: irb@wpi.edu
Human Protection Administrator: Gabriel Johnson, Tel. 508-831-4989, Email: gjohnson@wpi.edu

Title of Research Study: Researching the effects of latency on cloud games

Sponsor: Google

Introduction:

You are being asked to participate in a research study. Before you agree, however, you must be fully informed about the purpose of the study, the procedures to be followed, and any benefits, risks or discomfort that you may experience as a result of your participation. This form presents information about the study so that you may make a fully informed decision regarding your participation.

Purpose of study:

The purpose of the study is to see the effect that latency in video games has on a user's enjoyment and performance, and how this effect changes based on latency compensation as well as type of action the user does in the game.

Procedures:

We will have you, the player, play the game for a short duration before we ask you to fill out a survey on your satisfaction, as well as record your performance. We will then change the various variables such as latency or difficulty before having you play another short session. We will repeat this process until we have collected our needed data points. You are free to leave any time, and your participation is voluntary.

Risks:

While we will clean all surfaces and observe social distancing, there is a risk that you will catch the coronavirus from being in the study environment.

Benefits:

By participating in this study, you will be entered into a raffle for a Google Stadia access code. Also, for IMGD majors, this study will count towards your playtesting requirement.

Confidentiality:

"Records of your participation in this study will be held confidential so far as permitted by law. However, the study investigators, the sponsor or it's designee and, under certain circumstances, the Worcester Polytechnic Institute Institutional Review Board (WPI IRB) will be able to inspect and have access to confidential data that identify you by name. Any publication or presentation of the data will not identify you.

Compensation in case of injury:

You do not give up any of your legal rights by signing this statement

Your participation in this research is voluntary. Your refusal to participate will not result in any penalty to you or any loss of benefits to which you may otherwise be entitled. You may decide to stop participating in the research at any time without penalty or loss of other benefits. The project investigators retain the right to cancel or postpone the experimental procedures at any time they see fit.

By signing below, you acknowledge that you have been informed about and consent to be a participant in the study described above. Make sure that your questions are answered to your satisfaction before signing. You are entitled to retain a copy of this consent agreement.

Study Participant Signature Date:

Study Participant Name (Please print)

Section 9.2 IRB

Notification of IRB Approval

Date:	05-Nov-2020
PI:	Claypool, Mark L
Protocol Number:	IRB-21-0158
Protocol Title:	The Effects of Latency and Compensation on Game Mechanics
Approved Study Personnel:	Fan, Tian Yu~Xu, Xiaokun~Guan, Zijian~Carlson, Edward~Claypool, Mark L~
Effective Date:	05-Nov-2020
Exemption Category:	3
Sponsor*:	

The WPI Institutional Review Board (IRB) has reviewed the materials submitted with regard to the above-mentioned protocol.We have determined that this research is exempt from further IRB review under 45 CFR § 46.104 (d). For a detailed description of the categories of exempt research, please refer to the IRB website.

The study is approved indefinitely unless terminated sooner (in writing) by yourself or the WPI IRB. Amendments or changes to the research that might alter this specific approval must be submitted to the WPI IRB for review and may require a full IRB application in order for the research to continue. You are also required to report any adverse events with regard to your study subjects or their data.

Changes to the research which might affect its exempt status must be submitted to the WPI IRB for review and approval before such changes are put into practice. A full IRB application may be required in order for the research to continue.

Please contact the IRB at irb@wpi.edu if you have any questions.

Section 10. References

[1] Miyazu, H. “Cloud Gaming Market : Business Overview, Upcoming Trends and Top Company Analysis Forecast.” *MarketWatch*, MarketWatch, 5 Feb. 2021, www.marketwatch.com/press-release/cloud-gaming-market-business-overview-upcoming-trends-and-top-company-analysis-forecast-2021-02-05.

[2] Warren, T. “Google Unveils Stadia Cloud Gaming Service, Launches in 2019.” *The Verge*, The Verge, 19 Mar. 2019, www.theverge.com/2019/3/19/18271702/google-stadia-cloud-gaming-service-announcement-gdc-2019.

[3] Claypool, M. & Claypool, K. “Latency and Player Actions in Online Games,” *Commun. ACM*, vol. 49, pp. 40–45, Nov. 2006.

[4] Long, M., & Gutwin, C. “Effects of Local Latency on Game Pointing Devices and Game Pointing Tasks.”*Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, 4 May 2019, doi:10.1145/3290605.3300438.

[5] Rouse, M. “Thick Client (Fat Client).” *WhatIs*, TechTarget, 1 Sept. 2020, whatis.techtarget.com/definition/fat-client-thick-client.

[6] Forcepoint. “What Is a Thin Client?” *Forcepoint*, www.forcepoint.com/cyber-edu/thin-client.

[7] “Stadia GDC 2019 Gaming Announcement.” *YouTube*, 19 Mar. 2019, www.youtube.com/watch?v=nUih5C5rOrA&ab_channel=Google.

[8] Chen, K., Chang, Y., Tseng, P., Huang, C. & Lei, C. “Measuring the Latency of Cloud Gaming Systems.” *Proceedings of the 19th ACM International Conference on Multimedia*. Scottsdale Arizona, USA. Association for Computing Machinery

[9] Verizon. “Bandwidth.” *Verizon*, www.verizon.com/info/definitions/bandwidth/.

[10] Claypool, M. Claypool & Claypool, K. “Latency Can Kill: Precision and Deadline in Online Games,” *Proceedings of the First ACM Multimedia Systems Conference,* (Scottsdale, AZ, USA), February 2010.

[11] Dabrowski, R., Manuel, C. & Sieja, R. “The Effects of Latency on Player Performance and Experience in a Cloud Gaming System”.*Digital WPI IQP,* 7 May 2014.

[12] NVIDIA. “Introducing NVIDIA Reflex: Optimize and Measure Latency in Competitive Games.” *NVIDIA*, www.nvidia.com/en-us/geforce/news/reflex-low-latency-platform/.

[13] Lee, I., Kim, S. & Lee, B. “Geometrically Compensating Effect of End-to-End Latency in Moving-Target Selection Games.” *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, May 2019, https://dl.acm.org/doi/10.1145/3290605.3300790.

[14] Desveaux , A.C. & Courtemanche, V.S. “The Effects of Latency in Commercial Cloud Video Gaming Services.” *Digital WPI IQP*, 19 Mar. 2020.

[15] Anouna, J., Estep, Z. & French, M. “Network Latency and Cloud Games: A Study Using GamingAnywhere.” *Digital WPI IQP*, 4 May 2014.