Effects of Jitter on Quality of Experience in Cloud Gaming

Thomas Flanagan, Carter Nakagawa, Michael Oliveira
Professor Mark Claypool
March 3 2023

Abstract

The emergence of cloud technology has led to cloud-based video game streaming (cloud gaming for short), which provides access to games without expensive hardware. However, cloud gaming is susceptible to variations in latency that impact performance and user experience. A better understanding of these effects can help build systems to mitigate them. This project explores the impacts of jitter frequency and magnitude on user experience. Four games, Bloons Tower Defense 6, Hollow Knight, Hades, and CS:GO, were tested by having participants rate gameplay experience under various jitter conditions. From this experiment, jitter magnitude was found to have a significant, negative effect on user experience, while jitter frequency was found to be inconclusive.

Table of Contents

1. Introduction

While there are multiple benefits to cloud-based streaming that improve on the current standard of local gaming, latency can have significant negative impacts on gaming experience. As a first step at addressing this issue, many researchers study the effects of latency on player experience to understand which aspects need to be improved and how these improvements can be made (Claypool and Claypool, 2010). However, current literature on latency in cloud gaming and streaming overall has focused primarily on the effects of constant latency, also known as delay (Schmidt et al., 2017). Non-constant latency, known as jitter, is less understood and may have a separate and significant effect on the quality of cloud-based game streaming.

Previous studies have looked for relationships between network conditions and quality of experience. Studies into non-interactive video playback found the amount of delay with an equivalent impact on quality of experience as an interrupt in playback (Barman, 2018). Studies in interactive media have concluded the types of interaction that impact quality of experience the most when degraded (Claypool and Claypool, 2010). They have also found the effect of delay on quality of experience. However, there is little existing research on the effects of interrupts on quality of experience in interactive media.

Our main goal is to better understand the effects of jitter in the context of cloud gaming systems. The findings from this study on jitter can be used for the improvement of cloud gaming systems. Specifically, a better understanding of jitter can be used by playout buffer management to improve in-game experience. We accomplish this by measuring the effects of jitter on player experience through a study recording participants' response to jitter of varying frequency/magnitude.

This project explores the relationship between the different aspects of jitter and the quality of experience across a variety of game genres, visual complexities, and camera types. The first step of this process was the selection of games to be tested, decided by comparing game genres, camera types, and spatial/temporal information metrics (International Telecommunications Union, 2008). Following that, a streaming platform was selected that fit the studies requirements, including that the streaming software be open-source and support the selected games. Pilot studies were then done

to identify specific experimental conditions with varying jitter magnitude and frequency to be used in the experiment. Users were then recruited from the student body via email. These users played the selected games at all experimental conditions and rated their experience on a 1-5 quality of experience (QoE) scale. Performance metrics were also recorded through the use of the presentmon tool (Montgomery, 2022). Scripts were used to automate much of this process, keeping proctor interference to a reasonable minimum. In addition to QoE, demographic information about users was also recorded, including age, gender, and previous gaming experience.

Following the experiment, demographic analysis and data cleaning were carried out to characterize the data and ensure its quality. The rest of the analysis focused on the relation between jitter frequency/magnitude and QoE, first attempting to identify significant differences between experimental conditions. ANOVA tests identified significant differences between differing magnitude settings, but not conclusively between different frequency settings. Further analysis focused on modeling the relations between jitter and user experience. This analysis identified a negative relationship between jitter magnitude and average QoE, computed through kernel density estimates, suggesting that QoE decreases with increasing jitter magnitude in a roughly linear way. No such relationship was found for jitter frequency, though some tests suggested that it may have some effect, with average QoE showing a slight decrease as the frequency of interrupts increases.

The rest of this paper is organized as follows: Chapter 2 outlines background information needed to understand this project. Chapter 3 explains the methodology of the experiment, and how that methodology was developed. Chapter 4 describes the data gathered from the experiment, and analyzes it to find a relationship between jitter and quality of experience. Chapter 5 outlines our conclusions based on the data gathered. Chapter 6 describes possible future work related to this topic.

2. Background

This section describes the preliminary information needed for this project. This includes cloud gaming systems (2.1), the specifics of buffering/latency (2.2), and quantifying user experience (2.3).

2.1 Cloud Gaming Systems

“Cloud computing” refers to the practice of running an application on a remote server and interfacing with it using a client service, “cloud gaming” is much the same except that the application in question is a video game.

2.1.1 Structure of Cloud Gaming Systems

In a cloud gaming system, the client provides the necessary tools to interface with a video game–typically, this means a controller, a display, and some form of speakers. This client takes the user input from the controller and sends it over the internet to the host, which takes the input into account while running the game itself while simultaneously streaming the video and audio of its output back to the client. Ideally, this results in a seamless experience on the client end, where the audiovisual outputs respond as soon as physically possible to their inputs on the controller, as would appear to be the case if their device were running the game without interfacing with the cloud system.

2.1.2 Thick vs. Thin Client Architecture

Cloud gaming requires that some computations must be done by the client while others are done by the host. This leads to the description of some cloud systems as having a “thick” client architecture or a “thin” client architecture. In the case of thick client architectures, most of the computations are done on the client end, with the thickest possible architecture being the entire program running locally. The opposite is a thin client architecture, where most of the computing and data are handled by the server (J. Gaskin, 2011). Since a thicker client architecture would already require a more powerful machine like a PC or console that could run games locally, these are not the

targets for game streaming platforms. With thinner architectures being focused on, the inputs and audiovisual output need to be handled on the client end, while the server needs to interpret those inputs once they arrive, run the game, and also send the game data back to the client. Because this requires a comparatively large amount of data to be transmitted, latency has more significant impact than might be seen in the same conditions for cloud applications with thick client architectures.

2.1.3 Benefits of Cloud Gaming Systems

Cloud gaming is mostly sought after for the low barrier to entry on the part of the player: it streamlines the experience of getting started with playing a game as much as possible and reduces any requirements in memory or other hardware components. Additionally, it keeps things simple by making games compatible across platforms, the data being stored on the same server with the clients themselves being interchangeable. While it introduces issues regarding network latency, these are becoming less relevant due to the advent of 5G networks increasing the bandwidth available to a significant amount of the potential market as well as ushering in advancements in cloud computing architectures which can help to further optimize the performance of cloud gaming services (Shatzkamer, 2022).

2.1.4 Growth of Cloud Gaming

While cloud-based gaming services have been introduced by some companies already associated with the video game industry such as Microsoft and Nvidia in addition to big tech companies like Amazon and Facebook, their market share remains fairly small and accounts for only 6% of consumer spending on video games (K. Browning, 2021). In order for the sector to grow, cloud gaming services have to overcome several key hurdles, both technical and commercial. With regards to the technical hurdles, the main issues concern the necessity of a particularly strong internet connection and the extent to which that affects the overall game performance and user experience (S. Naji, D. Abrahams, 2022).

2.2 Buffering and Latency

Latency poses a significant issue for cloud gaming systems, as it can cause significant degradation in a user's gameplay experience. In this section, methods for characterizing latency and mitigating its negative effects in a cloud gaming system are described.

2.2.1 Delay

Delay is a measure of time for data to transfer between network locations. It is typically measured in milliseconds. Networks naturally have some amount of delay, due to needing to process and route data as well as the time the data takes to physically travel. The amount of delay on a local network is usually trivial compared to the reaction time of the average person (K. A. Rahman et al, 2019). When delay is significantly large, it can make a game feel less responsive. This is because the game takes longer to react to inputs, and can feel sluggish to control. However, humans can generally adapt to small amounts of delay by leading their inputs ahead of when they would normally enter them. For the purpose of our study, delay will refer to a constant amount of additional delay added to a local network.

2.2.2 Jitter

Jitter is a measure of an inconsistent amount of time taken for data to transfer between network locations. Rapid and unpredictable changes from jitter can disorient a player in a way that constant delay would not. A human can not easily adapt to random change in response time, making the effects of jitter a topic of interest. The individual periods of delay that characterize jitter are referred to as interrupts. Different frequency and magnitude of these interrupts leads to different forms of jitter. Differences in these forms of jitter are of particular interest.

2.2.3 Bitrate

Bitrate refers to the amount of bits in an amount of time that can be transferred over a connection. Generally measured in megabytes/second, it is a limiting factor in

video streaming. The maximum frame rate and resolution for a video stream is limited by the bitrate of the connection, as video with more frames or a higher resolution requires sending more bits to transfer the increased amount of data in the same amount of time. If the network connection of the client is not fast enough to match the desired rate of data transfer, the video stream cannot get all of the information it needs in real time, and sacrifices have to be made. Lowering the bitrate of the video stream to below the average speed of the network connection will restore a smooth playback, but results in lower visual quality.

2.2.4 Video Encoding and Compression

A fully uncompressed video file consisting of every pixel of every frame is impractical for most applications, including video streaming. On top of lossless compression algorithms which can help reduce the overall file size, video encoding formats exist which simplify how the video is stored to a point that humans cannot perceive the difference. If necessary due to file size/bandwidth constraints, the files can also be further compressed even if artifacts of the process are perceptible.

Compression is done by identifying redundant information and representing it using fewer bits. For video signals, redundancy is identified in four categories: spatial, temporal, perceptual, and statistical. Spatial redundancy is found when neighboring pixels within a single frame are similar, such as if there is a large area that is all the same color. Temporal redundancy occurs when neighboring frames are similar, such as if a large area of the picture does not change for several consecutive frames. Perceptual redundancy occurs due to quirks of the human eye, such as its difficulty discerning between slightly different colors of the same brightness. Statistical redundancy further improves the compression of the other redundancies to encode more frequently occurring image parameters with fewer bits than others.

Redundancy in audio signals is found in four categories: threshold of hearing, masking, stereophonic irrelevance, and statistical redundancy. Redundancy in the threshold of hearing is found due to the fact that the dynamic range between the threshold of hearing and threshold of pain for the volume of audio differs depending on frequency, particularly in that the highest and lowest frequencies audible to the human

ear do not need as much resolution to be faithfully reproduced. “Critical bands” refer to bands of frequencies that the human ear can distinguish between, forming a perceptual pitch scale. A loud frequency being played reduces the dynamic range of hearing for the rest of its critical band, meaning a technique called frequency masking can be applied where quiet enough frequencies within that band do not need to be encoded. Similarly, once the masking frequency is removed, the masking effect can be gradually removed over the course of about .2 seconds, and when introduced it can fade in for about 50 ms; this is known as temporal masking. Stereophonic irrelevance refers to the fact that the human ear can not perceive stereo effects at lower frequencies, so those can be encoded as mono (Austerberry, 2005).

2.2.5 Sources of Latency

Latency is a measurement of the amount of time it takes some input to produce some output in a machine. Latency is typically measured in milliseconds. There are many sources of latency when using a computer, and some sources are easier to mitigate than others. Between input from a mouse or keyboard, the machine processing input, the HDMI connection transferring a frame, and output from the monitor, a typical computer interaction has up to 50ms of latency due to hardware limitations (K. A. Rahman et al, 2019). The issue of latency is confounded when data is transmitted between machines over a network. Cloud gaming operates by reading user input on the client machine, sending that input over a network to the server machine, processing that input on the server machine, and sending rendered frames back to the client machine to display. On top of the traditional sources of latency when playing a game, sending and receiving information over a network adds a potentially significant amount of additional latency. If a client’s network connection is slow or unstable, sacrifices can be made in at least one aspect of the stream in order to compensate for the network limitations. These options include: bitrate, framerate, and buffer size.

2.2.6 Reducing Visual Quality

When a client does not have enough bandwidth to support a stream, interrupts will occur, as the playback cannot fetch frames fast enough to produce a smooth output.

Lowering the frame rate or bitrate of the stream reduces the amount of bandwidth required to keep that stream smooth. This reduces the visual quality, but is unavoidable when the network connection simply does not have the bandwidth for a higher quality stream. Most cloud gaming platforms automatically reduce the frame rate or bitrate when detecting a low bandwidth connection. Our experiment attempts to avoid changes in the visual quality of the stream, in order to avoid confounding variables when measuring the quality of experience.

2.2.7 Playout Buffers

When a client experiences high or unstable latency between sending a packet of inputs and receiving frames of video, frames can be buffered to improve the stability and smoothness of playback despite frames arriving after large and varying amounts of time. While having a larger buffer size allows for more leeway in the time a packet arrives, it can impact the quality of experience for the player by making the game harder to properly control. The alternative option is to allow interrupts to happen, and accept playback which is not smooth. For non-interactive video streams, interrupts have a much more significant effect on the quality of experience, so larger delays are more acceptable if they prevent interrupts. Figure 1 shows the relationship between interrupts, buffering, and quality of experience (J. Allard et al, 2020). About 30 seconds of buffering is equal in annoyance as about two interrupts during playback.

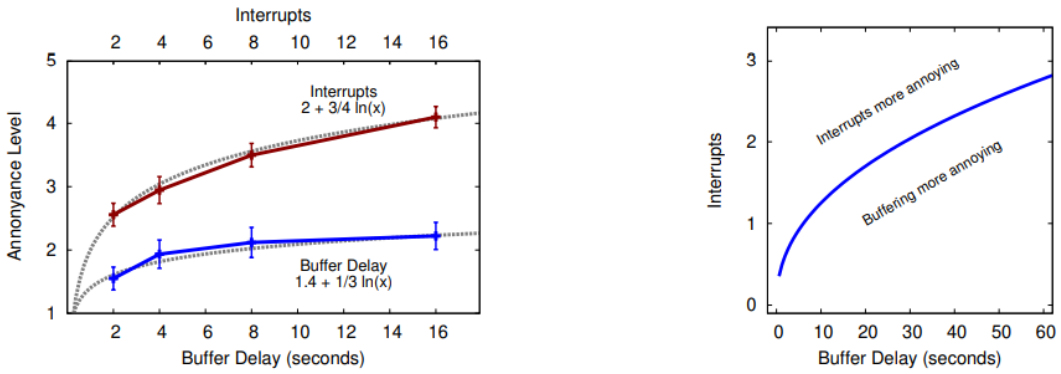


Figure 1: Amount of annoyance with increasing delay time and number of interrupts for non-interactive video. Approximation of an equivalence line between interrupts and buffer delay. (J. Allard et al, 2020)

The interactivity of cloud gaming means that buffering impacts quality of experience at much smaller buffer sizes than in non-interactive video. Rather than taking two to four seconds of delay to noticeably impact quality of experience, less than 500ms of delay impacts quality of experience (S. Schmidt et al, 2017). An important part of developing a cloud gaming platform is figuring out how to balance the line between allowing interrupts versus buffering playback and adding additional delay.

2.3 User Experience

The goal of improving user experience is a main motivator in cloud gaming research. Understanding the effects of latency in a cloud gaming system from a user's perspective can help identify aspects that matter most. This section outlines various methods to quantify and characterize gameplay experience.

2.3.1 Measuring Quality of Experience (QoE)

In order to investigate the effects of latency on cloud gaming services, different metrics are used to measure the in-game quality of experience (QoE). These metrics may be subjective or objective, with both types revealing different aspects of the gaming experience. Some measures aim to characterize games to show how different categories of games respond differently to latency. Other measures are concerned with the visual fidelity of a game as it is streamed to the client. Player opinion is also an important factor, so player ratings of game playability or general quality are also considered.

However, not all metrics are considered equally important or useful depending on the context of a given study. For our study, we are focused on metrics that measure user experience, so we would prioritize metrics concerned with player experience and gameplay type rather than measures of visual quality. Additionally, certain measures may be more strongly correlated to gameplay quality than others. For example, previous research has shown player rating of playability to be a more significant metric than visual quality when estimating overall QoE (A. Wahab et al., 2021). We also will take

into account how these metrics respond to latency. Existing literature describes multiple effects of latency on overall QoE, such as research by Schmidt et al. suggesting that changes in game “pace” affects QoE much more than changes in visual perspective (S. Schmidt et al., 2017).

2.3.2 Effect of Latency on Gaming Experience

To understand the relationship between latency and gaming experience, we have to also understand the ways in which latency manifests in-game. As mentioned above, latency in cloud gaming systems is a result of the time it takes for data to be transferred between client and server. While latency affects all game data that has to go through this network connection, it is more and less perceptible by the player depending on the context and game aspect. When looking at the ways a player may perceive latency, we broadly categorize such effects as input or video delay based on objective and subjective differences.

Input delay refers to the time between a player inputting a command and the result of the command appearing in the video stream. From a technical perspective, this effect is distinct because the degree of delay depends on the round-trip time from the client to the server and back. This is because the command is inputted client-side, sent to the server and processed, then affects the video stream sent back to the client. As a result, this kind of latency is often more perceptible than “one-way” latency, as the total delay will always be greater. This type of latency can also be more disorienting, as it makes it difficult for players to control a game accurately.

Video delay refers to the time it takes for the video stream of a game to be transferred from the host to the client. This is affected by the one-way delay from host to client, so this delay will generally be less than input delay. Video delay is a part of input delay but can also exist without player input (e.g. delay during a cutscene). While it may not have as much of an effect on player performance, it can still make gameplay unpleasant for a player if the effect is perceptible.

Both of these effects can vary in perceptibility and significance with changes in latency. For example, relatively constant latency may not have as much of an effect on

player experience as jitter. Additionally, constant latency at low levels may be imperceptible, but at higher levels may significantly impact gameplay.

2.3.3 Spatial Information/Temporal Information

One way we can categorize games is by their visual complexity. A standard method for this is to measure the spatial information (SI) and temporal information (TI) for a representative video recording of a game (International Telecommunications Union, 2008). Both values are calculated per frame, with the maximum values across all frames representing the SI and TI for a given recording.

In general terms, SI is calculated by comparing the brightness of each pixel in the filtered image of a given frame. The variation in brightness across all pixels is what determines the SI value for that frame. Games with high contrast visuals will therefore tend to have higher SI values.

The TI value is computed by comparing the change in brightness for each pixel in a given frame from its brightness in the previous frame. As with SI, the variation of the change in brightness for all pixels in a given frame is what determines the final TI value. Games with sudden visual changes occurring irregularly across the screen will tend to have higher TI values.

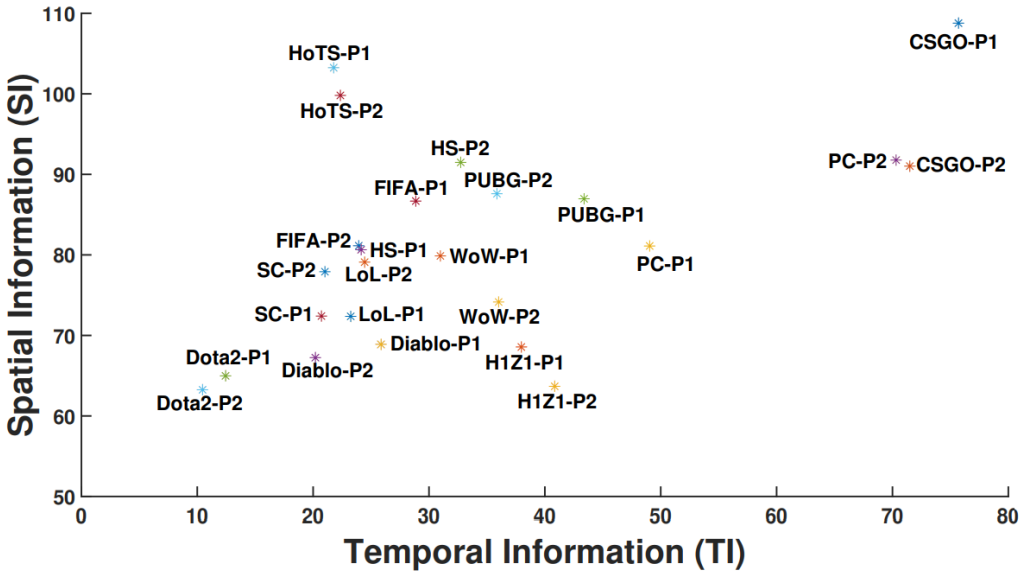


Figure 2: SI/TI values for game clips measured by Barman et al., two clips included per game (N. Barman et al., 2018).

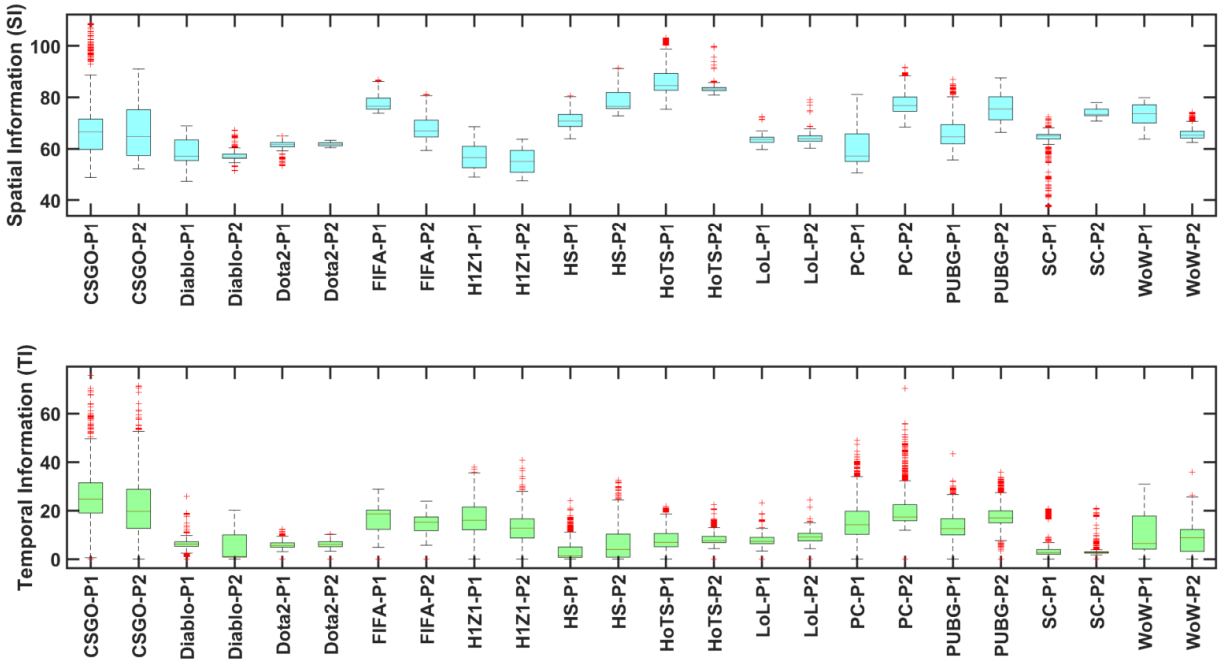


Figure 3: Distribution of frame-by-frame SI and TI values for each game clip from the study by Barman et al., visualized as box plots (N. Barman et al., 2018).

Previous studies have investigated the relationship between SI/TI values and subjective video quality. In the context of video game footage, Barman et al. found that the SI/TI for a game was related to the game's perceived visual quality when the footage was distorted (Barman et al., 2018). For games with moderate SI/TI, study participants reported similar levels of perceived quality at similar levels of distortion, controlled by changing the bitrate and resolution of pre-recorded footage. Games at SI or TI extremes, whether low or high, had much more variable perceived quality. This research also showed that SI/TI was a more important metric than game genre in predicting perceived quality of game visuals.

2.3.4 Measuring Visual Fidelity

For cloud gaming systems, another area of interest is the fidelity of video streams over the connection between host and client. Measuring how faithfully the image is reproduced client-side can be done through multiple metrics, including peak signal-to-noise ratio (PSNR) and structural similarity index measure (SSIM). PSNR is defined as the ratio of a signal's power to the power of any distorting noise. In the case of cloud gaming or other streaming services, it depends on the quality of the connection between host and server. SSIM, first introduced in 2004 by Wang et al., predicts the perceived image quality of a compressed image by comparing properties of the original and the compressed image (Z. Wang et al., 2004). Unlike PSNR, SSIM takes into account factors such as luminance and contrast masking that can better account for how the human eye perceives image quality. The two metrics are correlated and can predict each other but differ in sensitivity to specific kinds of image degradation (A. Horé and D. Ziou, 2010). While some research suggests that PSNR is a less reliable metric across more varied encoding settings and video content type, it remains a reliable and simple metric when applied with specific context in mind (Q. Huynh-Thu and M. Ghanbari, 2012).

2.3.5 Deadline/Precision

Many of the metrics discussed so far are concerned with video quality and apply to streaming services in general. However, it is also important to consider qualities unique to gaming services. The deadline/precision model proposed by Claypool and Claypool is a suitable option, focusing on player actions and input latency (Claypool and Claypool, 2006). This model considers the deadline and precision for individual player actions throughout gameplay. In this context, deadline is defined as the length of time a player may take to perform an action, while precision is defined as the accuracy required to successfully accomplish an action. Both measures are subjective, though objective measures of game mechanics may be considered to establish relative deadline/precision between actions (e.g. measuring target size or gameplay speed).

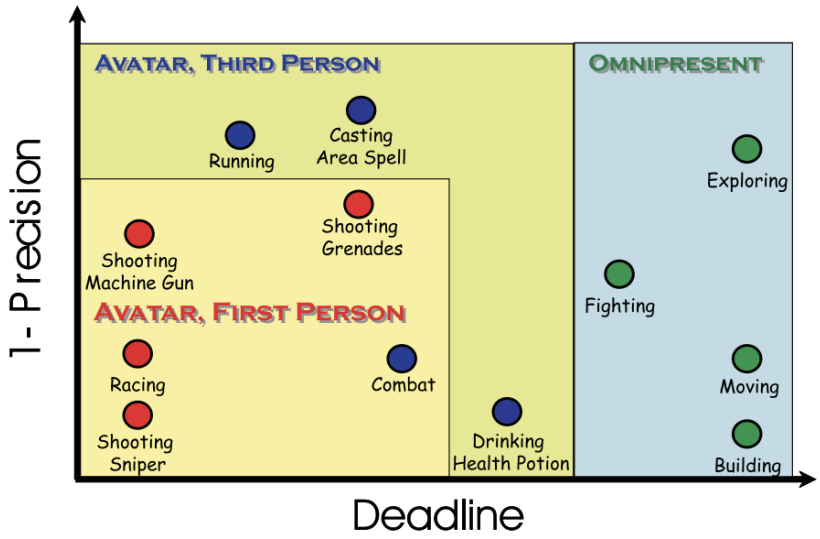


Figure 4: Examples of game actions categorized by genre visualized in the deadline/precision space (M. Claypool and K. Claypool, 2006).

In the study by Claypool and Claypool, deadline and precision were shown to be significant in predicting the effects of latency on player experience. The study modified mechanics of a 3rd person shooter game, including avatar size and projectile speed, in order to vary deadline and precision. The researchers also controlled latency between the host and client machines. Player performance, including score and other metrics,

were used to measure in-game experience. The results showed that games with high precision and tight deadlines tend to be more negatively affected by latency. With such conditions, player inputs often result in unexpected outcomes, which negatively affects performance.

In our research, we consider the deadline/precision model to be a useful addition to other metrics, particularly for its relevance to player experience specifically. As discussed previously, input lag is an important aspect of latency in cloud gaming services that is not accounted for by other metrics.

3 Methodology

This section explains the process of creating a methodology for this experiment. It presents the process used to select games for the experiment (Section 3.1) and how the experiment was developed (Section 3.2).

3.1 Game Selection

This section outlines how games were classified before selection (Section 3.1.1) and how games were selected to be used in the experiment (Section 3.1.2).

3.1.1 Game Classification

Previous studies suggest differences in perceived QoE adding delay to a game can depend on spatial/temporal complexity. In order to study the effects of jitter on a wide variety of spatial and temporal complexities, we used a number of different criteria to classify a pool of candidate games. The pool was built from games owned by our IQP team and the adjacent MQP team also working with Google Stadia (R. Darcey et al., 2022). A total of 38 unique games were included in this initial assessment (Table 1). For each game, either we or the MQP team captured 30 seconds of generic gameplay at 1080p resolution and 60 frames per second and used that footage to determine a maximum SI and TI value for that game. The footage from the 16 games tested by us was also used to determine average SI/TI, median SI/TI, and the standard deviation of SI/TI for those games. Both personal and laboratory devices were used to collect this initial footage. The camera type of the game was also recorded, and was marked as one of “Fixed”, “2D side view”, “2D top-down view”, “3D first person”, or “3D third person”. From this pool, we selected four games which covered a range of SI/TI values and a range of camera types. We aim to choose games across a range of SI/TI values to verify significant differences are also present when adding jitter. We also hypothesize that the camera perspective of a game adds complexity to game visuals and the overall experience that is not captured by SI/TI alone.

Table 1: All games included in the initial search for games.

Game Name	Camera Type	SI	TI
Ages of Empire	2D Top	116.319	40.852
Assassin's Creed Origins	3D Third Person	78.656	63.227
Bloons Tower Defense 6	Fixed	59.270	21.159
Celeste	2D Side	65.376	42.530
Clone Hero	Fixed	43.746	19.874
CS:GO	3D First Person	40.065	30.728
Cuphead	2D Side	52.272	64.680
Deep Rock Galactic	3D First Person	52.090	46.893
Don’t Starve	2D Top	45.170	10.762
Doom	3D First Person	65.614	71.608
Dragon Ball FighterZ	2D Side	143.123	96.685
Factorio	2D Top	68.621	30.117
Fall Guys	3D Third Person	63.158	59.696
Grand Theft Auto V	3D Third Person	83.597	43.313
Hades	2D Top	94.938	72.525
Hollow Knight	2D Side	41.646	25.829
Katamari Damacy REROLL	3D Third Person	58.664	37.965
Kenshi	2D Top	94.004	63.036
League of Legends	2D Top	88.538	36.586
Lego Star Wars: The Skywalker Saga	3D Third Person	88.897	56.178
Minecraft	3D First Person	45.215	34.107
Minecraft	3D First Person	90.600	64.284
Octopath	2D Side	109.810	65.814
Osu	Fixed	78.502	37.068
Portal	3D First Person	68.658	56.801
PORTAL 2	3D First Person	31.638	29.623
Republic Commando	3D First Person	53.766	76.215
Rocket League	3D Third Person	63.943	41.310
Satisfactory	3D First Person	66.993	50.991
Sims 4	2D Top	59.524	45.183
Skyrim	3D First Person	53.923	29.892
Skyrim (First Person)	3D First Person	109.256	46.421
Skyrim (Third Person)	3D Third Person	108.316	36.511

Slay the Spire	Fixed	53.725	72.303
SPORE	2D Top	30.187	5.534
Stellaris	2D Top	80.966	40.239
The Return of the Obra Dinn	3D First Person	195.080	94.119
The Talos Principle	3D First Person	57.179	41.020
The Witcher 3	3D Third Person	52.040	31.750
The Witcher 3	3D Third Person	87.410	59.923
World of Horror	Fixed	103.370	30.397

Figure 5: Scatterplot of games by SI/TI values. Points are labeled with colors based on camera-type as shown by the legend.

Figure 5 shows the results of SI/TI analysis for the 41 games collected by the IQP and MQP teams. While the majority of games, regardless of camera type, fall around the center of the distribution for both metrics, there are some outliers. Examples of outliers include Spore on the low end of both SI and TI and DragonBall FighterZ on the high end of both SI and TI.

3.1.2 Selection Criteria

In order to narrow down the initial pool of games, some additional criteria were considered. Games with unavoidable exploratory or open-ended gameplay were

excluded, in order to keep the gameplay consistent between participants and trials. Games with a steep learning curve were excluded to reduce the amount of time needed to spend explaining the game and having the user practice. A player who does not feel like they understand how to play a game may also react more negatively to the game as a whole, which would affect their reported QoE. Although variations in skill were expected, games that are easier to learn were prioritized.

Within these additional restrictions, we also wanted to find games which covered a range of camera types and SI/TI values. We graphed the SI and TI over time for games we considered at this point, and tested out playing the games with different jitter settings to get a feel for how they would play. Four games were eventually selected that fit this goal, as shown in Table 2. Example screenshots from each game are shown in Figure 6. The SI and TI over time for each game selected is shown in Figure 7.

Table 2: Selected games and their considered criteria.

Title	**Publisher**	**Release Year**	**SI**	**TI**	**Camera Type**
Bloons Tower Defense 6	Ninja Kiwi	2018	Medium	Low	Fixed
Hollow Knight	Team Cherry	2017	Medium	Medium	2D Side
Hades	Supergiant Games	2018	High	High	2D Top-Down
Counter-Strike: Global Offensive	Valve Corporation	2012	Medium	Medium/ High	3D First-person

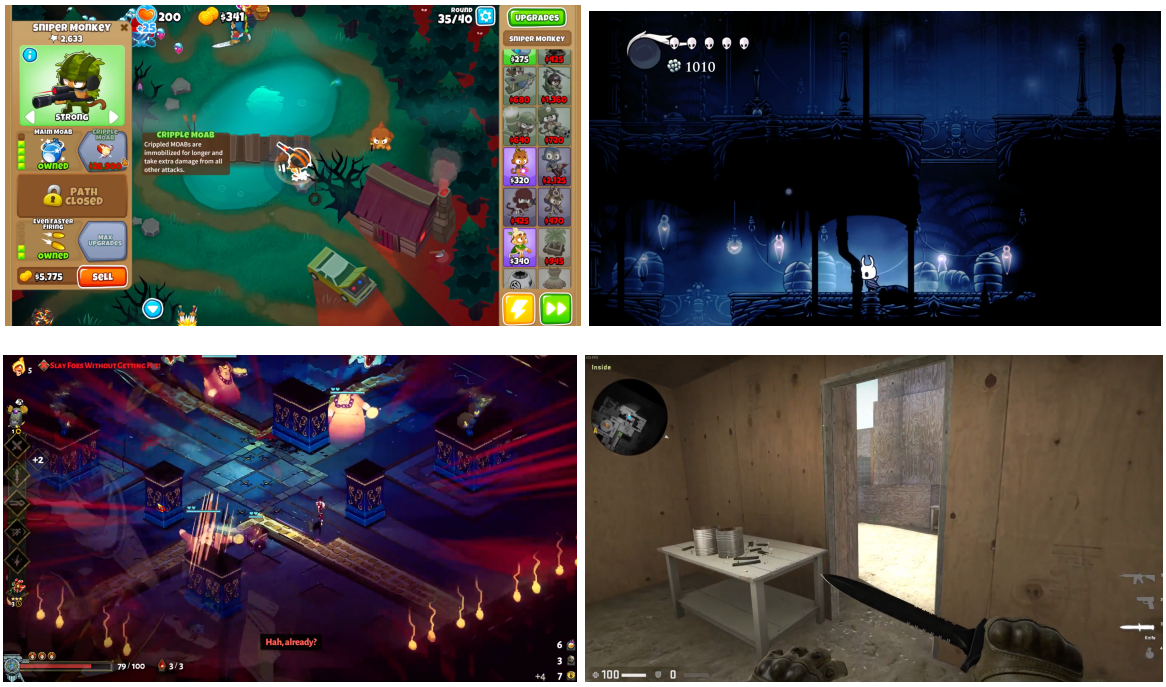


Figure 6: Example screenshots from Bloons TD6 (top-left), Hollow Knight (top-right), Hades (bottom-left), CS:GO (bottom-right).

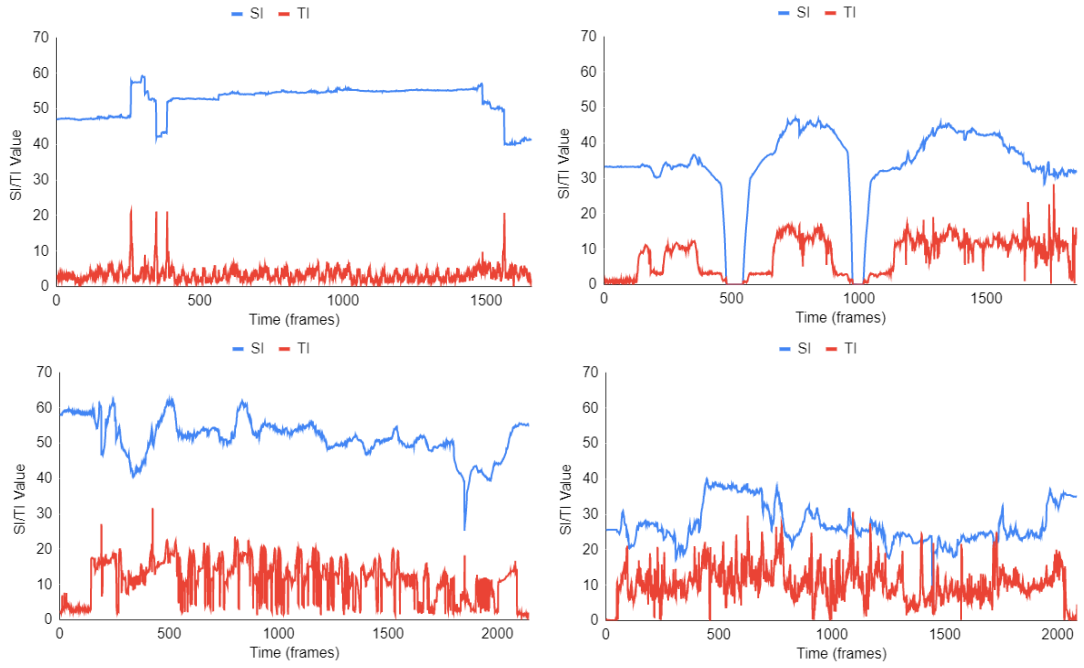


Figure 7: The SI and TI values over time for the four games Bloons TD6 (top-left), Hollow Knight (top-right), Hades (bottom-left), CS:GO (bottom-right)

Bloons Tower Defense 6 (BTD6) is a tower defense game about placing monkeys that pop balloons before they can travel across a track. It features a fixed camera, with little amounts of visual change over time. For our experiment, the map Monkey Meadow was played on Medium difficulty.

Hollow Knight is a metroidvania game featuring basic platforming, combat, and exploration mechanics. It features a 2D camera giving a side view of the environment, and a medium amount of visual change over time. For our experiment, a fresh save file was played from the beginning through the linear tutorial section of the game, which mostly consisted of platforming challenges and a small number of enemies near the end.

Hades is a roguelike game about fighting waves of enemies in randomly ordered rooms, growing stronger with upgrades at the end of each room. It features a 2D isometric camera looking down at a character who can move in 4 directions, and a high amount of visual change over time. For our experiment, a fresh save file was played from the beginning with the default sword weapon, with participants being directed to attempt a new run if they died.

Counter-Strike: Global Offensive (CS:GO) is a first person shooter which primarily focuses on tight aiming controls and a fast time to kill. It features a 3D first person camera. Though it only has a medium amount of visual change over time, this change is exacerbated by the player directly controlling the character with their mouse, rather than simply following their actions. For our experiment, the training course was played from start to finish, which features time based shooting challenges but no actual human or NPC enemies.

3.2 Study Implementation

This section outlines how a streaming platform was selected for the experiment (Section 3.2.1), how the initial design of the experiment was created and tested (Section 3.2.2), how the script used by proctors during the experiment was developed (Section 3.2.3), the process of obtaining participants for the experiment (Section 3.2.4), how the procedure was changed from the initial design over the course of the study (Section 3.2.5), and the final procedure used for the study (Section 3.2.6).

3.2.1 Platform Selection

Members of both the MQP and IQP teams researched 7 different platforms for hosts and clients: GamingAnywhere, Moonlight, Sunshine, OpenStream, Parsec, Steam Link, and Rainway. This was done in order to find a platform that fit the needs of the study. For the study, it was necessary that the platform be open-source, have access to a wide selection of games, and run with no significant latency. This was done to allow flexibility in the study design and to prevent any latency from the platform itself from interfering with the experiment. One other requirement was that the platform would not compensate for latency through any sort of scaling (e.g. lowering the image resolution). Based on these criteria, the platform Moonlight was chosen for the client and OpenStream was chosen for the host.

Moonlight is a free, open-source implementation of NVIDIA's GameStream protocol (Bergeron et al., 2022). It is able to stream any PC game from a machine compatible with the original GameStream protocol. Openstream is a free, open-source hosting platform chosen for its compatibility with Moonlight in addition to previously mentioned specifications (Openstream Team, 2020). Openstream was chosen over the similar, Moonlight-compatible hosting service Sunshine due to ease-of-use and overall quality.

The machines designated for the study were then configured with the streaming software installed. The client machine, with Moonlight installed, was connected to the host machine, with Openstream installed, through a router. The purpose of the router was to control the latency in the host-client connection using NetEm, an extension of built-in Linux traffic control capabilities (Hemminger et al., 2011).

While most of the criteria was verified by reviewing available documentation, it was necessary to manually check for any scaling done by the platform. This was done by running the platform on the host/client system and verifying visually that there were no scaling effects present when jitter was added. The video stream on the client machine was recorded so that segments with added jitter could be compared to baseline conditions.

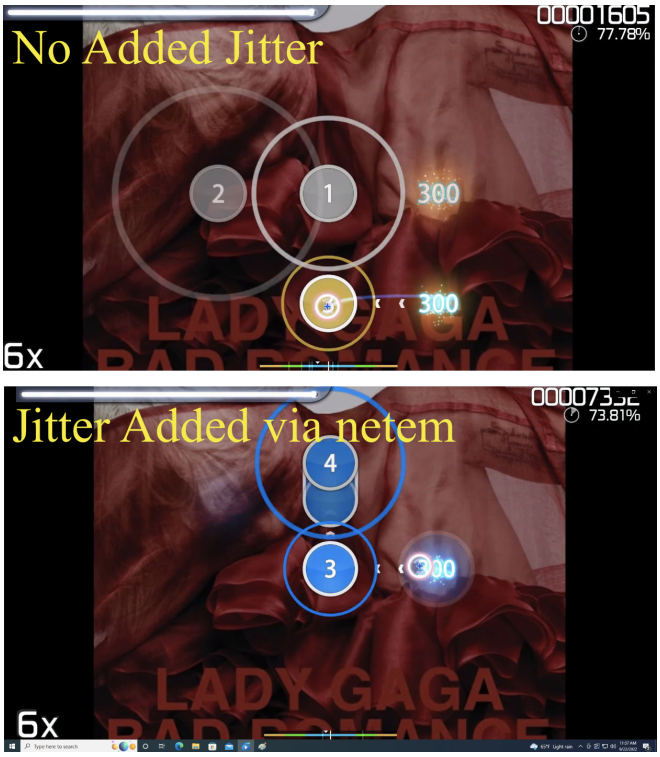


Figure 8: Screenshots for the game Osu run at 60 fps on the client machine with no added jitter (top) and added jitter through NetEm (bottom).

Figure 8 shows screenshots from the client machine during this test. The game Osu was chosen for this test because changes in resolution would be noticeable, as the game graphics do not vary much throughout gameplay. No difference in visual quality was observed between the two conditions.

3.2.2 Pilot Study/Initial Design

Following platform and game selection, initial study design began. At this point, the broad outline of the study was as follows:

- Participant plays four different games
- For each game, the participant plays for a fixed time at varying settings of jitter frequency/magnitude
- For each of these trials, the participant rates their quality of experience (QoE)

From this broad design, steps were taken to work out specific details. Firstly, the effectiveness of the tool NetEm for accurately controlling jitter across the client/host connection was verified. This was then extended to in-game tests, which were also used to determine appropriate levels of jitter to use in the study. All four games chosen for the study were tested at varying levels of jitter frequency/magnitude to identify settings where game performance issues were barely noticeable, notable but tolerable, and intolerable to the point of unplayability. These three settings were determined for both frequency and magnitude independently, resulting in 9 different frequency/magnitude combinations to test, along with the control condition (no jitter added). The machine participants would play on would have a 1080p display and display at 60 fps. The settings for each game would be adjusted to these settings. Each participant would play the game for 60 seconds per trial, with 10 trials of randomly chosen settings per game, and all four games played per person. For each trial, the participant would record their QoE for that trial.

In conjunction with the technical components of the study, a survey collecting demographic data as well as participant evaluations of in-game quality of experience (QoE) was designed. Demographic questions identified participant age and gender, and previous gaming experience. QoE questions asked participants to rate the gameplay on a 1-5 scale, with the option to provide decimal values. The final version of the questionnaire is located in the appendix.

These specifications were tested through a pilot study with IQP team members as both participants and proctors. This was done to not only confirm that the settings and procedure made sense, but also to gauge the length of the study. As the intended duration of the study was one hour, it was necessary to ensure that the total time of the trials and any other parts of the study would not exceed that window.

3.2.3 Proctoring Script Design

To ensure consistency throughout the study, Python scripts were used to automate some parts of the process. These scripts can be found in the appendix.

One script was run on the machine acting as a router which randomly selected the order of games and the order of trials within each game. Once a trial was started, it would automatically apply the correct NetEm settings for that trial, and would alert when the trial had ended. It also recorded the absolute time each trial took place at. This script outputs a CSV file for each participant with the randomly selected order of trials and information about those trials. This file was named according to a random participant identifier selected by the proctor.

The other script used was run on the client machine in order to collect data on the video stream. It launched a process of presentmon (Montgomery, 2022), which was used to track the amount of time between frames for the Moonlight video stream. It recorded the absolute time it began collecting data, to be matched up with the output of the router machine script. Starting and stopping the script was handled by the proctor to allow flexibility on a participant by participant basis. It outputs a file that was named according to the same random participant identifier selected by the proctor for the other script.

Many aspects of the study were not handled by script and left for the proctor to handle. The proctor was responsible for setting up each game up until the segment intended for the participant to play. It was decided that automating this through a script would be unnecessary and inefficient, as each game would require a different sequence of commands to reach the intended gameplay point. The proctor also was responsible for telling the participant when to pause the game. Pausing was not automated so as not to disorient the participant with a sudden loss of control. While this resulted in trials going slightly over the fixed time due to delays in participants pausing, the additional time was not considered in data processing, with only metrics from the fixed window being considered. In the pilot study, it was also determined that additional time from having the participant pause the game manually was not significant enough to be a concern.

3.2.4 Recruiting Participants

Participants were recruited via an email sent to CS and IMGD undergraduates at Worcester Polytechnic Institute. The email invited students to sign up for a 1-hour time

slot, with 56 available times over a two-week period. From this initial recruitment phase, 39 participants signed up. Participants were informed that they would be eligible for compensation in the form of a $10 Amazon gift card and/or IMGD playtesting credit if applicable. This email is included in the appendix.

3.2.5 Adjustments to Initial Design

Over the course of the study, it was necessary to adjust certain elements for efficiency and data quality. Firstly, the trial time was decreased from 60 to 50 seconds. This was done after three participants had completed the study, each taking longer than the expected 1-hour total time. Certain elements of the study were more time-consuming than initially thought (e.g. signing informed consent forms), so it was necessary to save time in other parts of the study. Changing the trial time from 60 to 50 seconds was considered a minimal change, as it would still provide a meaningful amount of data per trial.

Additionally, the proctoring policy was changed to allow proctors to guide participants if they were stuck at a certain point in the game. This guidance from proctors was infrequent, occurring 2-3 at most across a participant's entire trial and only in particular problem areas. This was especially important for games where the gameplay segment was non-repetitive, such as the tutorial segment in CS:GO and the post-death navigation back to gameplay in Hades. There were concerns that some participants getting stuck and missing gameplay that other participants experienced would have a significant effect on their overall experience. Therefore, it was deemed necessary to have proctors give gameplay advice to participants when necessary for them to progress.

3.2.6 Final Procedure

Participants were asked to arrive at a computer lab on campus for the experiment. Each participant was given an identifier that was used to combine their data later. The lab was arranged as shown in Figure 9, with the server and router machine back to back with the client and form machine. Each machine needed to be set up to an

initial state for each participant. The server and client machine were connected via the router machine to the same network. On the router machine, the proctor would launch the router machine script with the participant's identifier, which would direct the proctor on which game needed to be set up first. On the server machine, the proctor would launch OpenStream on the server machine, and then use Steam to launch the game identified by the router machine script. On the client machine, the proctor would launch the client machine script with the participant's identifier. The proctor would then launch Moonlight and connect the client machine to the server machine. On the form machine, the proctor would open the Google Sheets form with the participant's identifier. The proctor would also open the Game Tutorials document found in the appendix for the participant's reference.

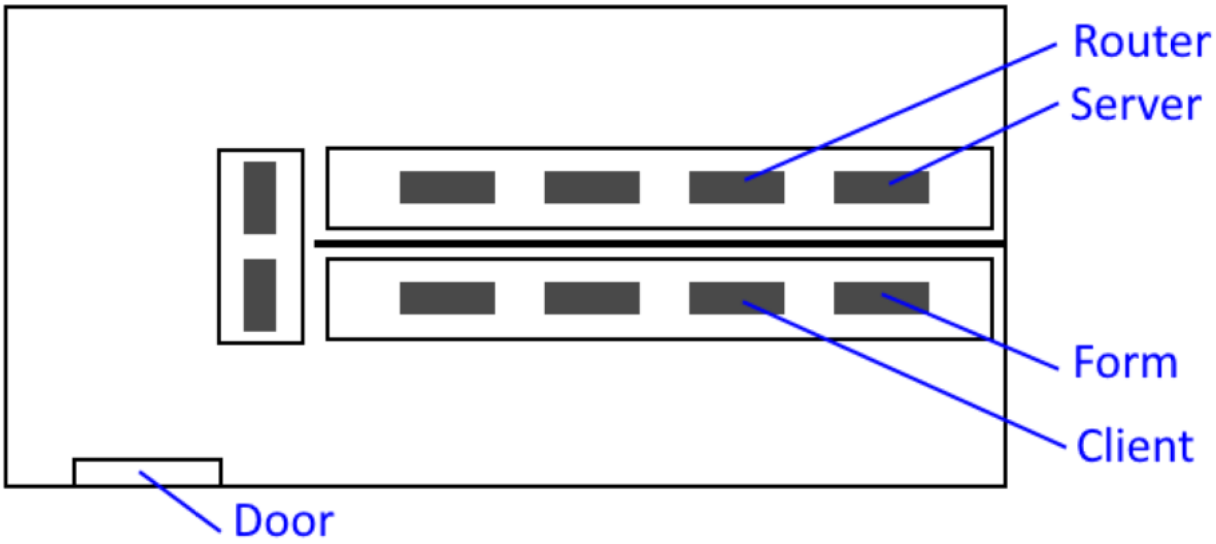


Figure 9: Layout of the lab. The computers used in the experiment are marked.

When the participant arrived, the proctor would ask them to sign the consent form found in the appendix. The proctor would then seat the participant at the client and form machines, and would ask them to fill out the demographic questions on the form machine. Once the participant was done, the proctor would ask the participant to practice the basic controls of the first game they would play. The participant was also directed to view the game tutorials document for assistance in learning the controls of

the game. After about a minute of practice, the proctor would tell the participant to stop and would begin the trials.

Each trial was started with a countdown for the participant to unpause the game. At the same time, the proctor would hit “enter” on the router machine script, which would start the trial. When the script outputs “STOP”, the proctor would ask the participant to pause the game and fill out an entry on the form machine. This would repeat for ten trials per game, and for four games in total. At the end, participants could optionally enter their email into the form in order to receive compensation of IMGD playtesting credit and a $15 Amazon gift card.

A simplified outline of the procedure is provided below for convenience. The steps are the same as outlined in the above description, only in the form of an ordered list.

1. Participant Arrival
    a. The participant fills out the informed consent form (see Appendix).
    b. The participant fills out the demographic survey (see Appendix).
    c. The participant is also asked to confirm they have had no symptoms of COVID-19 or contact with anyone who has.
2. Trial Period
    a. Game is selected randomly by the script and set up by the proctor on the host machine.
    b. The participant is asked to read the instruction manual for the current game (see Appendix).
    c. The participant is asked whether they require a practice period to learn the game controls. If so, a 1-2 minutes practice period is allowed as needed.
    d. One of ten jitter magnitude/frequency settings is set randomly by the script and the participant is instructed to begin playing.
    e. Once the trial time has elapsed, the script informs the proctor who then instructs the participant to pause.
    f. The participant logs their perceived QoE through the survey .
    g. Steps d-f are repeated nine more times until all ten settings have been tested.

    h. Steps a-g are repeated three times until all four games have been tested.

3. Post-trial Procedure
    a. The participant is asked if they would like compensation in the form of IMGD playtesting credit and a $15 Amazon gift card. If so, the participant logs their email in the form.
    b. The proctor asks the participant if they have any questions about the study and answers them if possible.

4. Results

This section analyzes the results from the experiment described in Section 3. This includes the demographic analysis (4.1), an overview of the results (4.2), descriptions of data cleaning steps (4.3), and statistical analysis of the data (4.4, 4.5).

4.1 Demographics

Twenty-eight people participated in this study. Fourteen identified as male, twelve identified as female, one identified as agender, and one did not respond. Ages ranged from 18 to 22 years, with a median of 19 and a mean of 19.5. The distribution of participant age is shown in the box-plot below (Figure 10). The red triangle marks the mean, the red line marks the median, and the circles denote outliers.

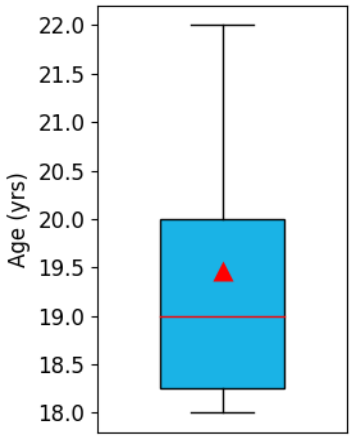


Figure 10: Distribution of participant age

The mean, median, and standard deviation for participant gaming experience, both general and game-specific, are listed in Table 3 below. Experience was rated on a 1-5 scale (1 = “very unfamiliar”, 5 = “very familiar”). Distributions of general and game-specific experience ratings are shown in Figure 11. The median self-rating for experience for each game was 2 for all games except for Hades, which was 1. The average experience level for all four games is lower than the average overall gaming experience level. This suggests that a significant portion of participants had not played these games in the past. Hades had the greatest number of participants who rated their experience as “Very unfamiliar”, and only one participant who rated it as “Very familiar”.

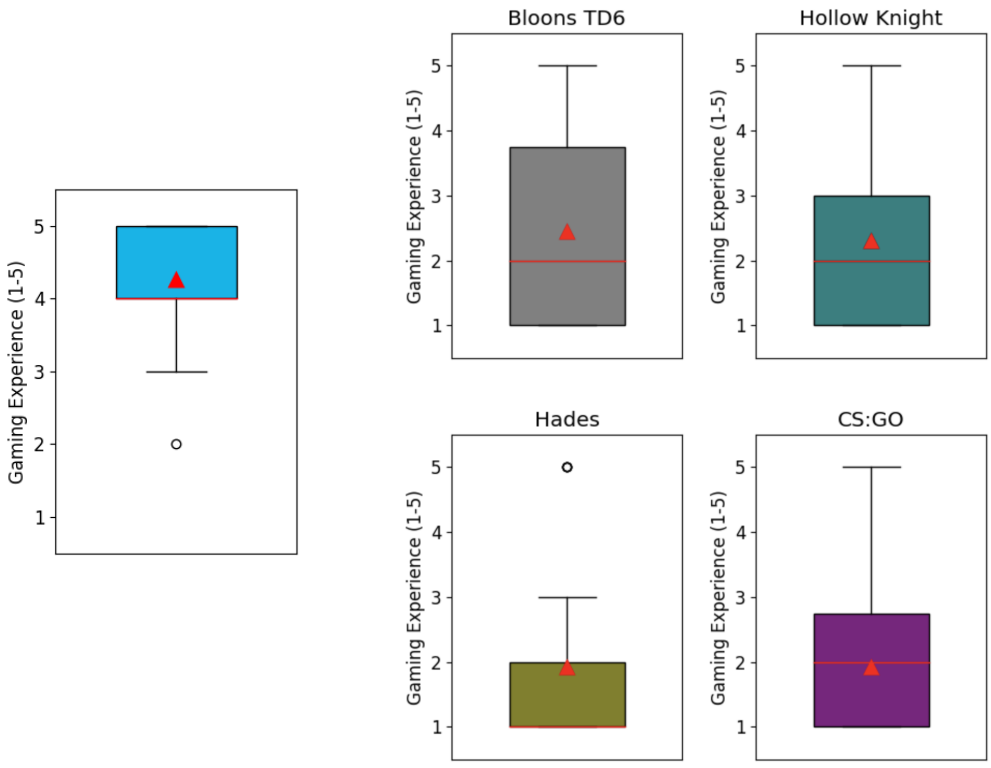


Figure 11: Distribution of participant gaming experience overall (left) and by game (right)

Table 3: Metrics of center and variance for all demographic measures.

Demographic Metric	Mean	Median	Standard Deviation
Age (years)	19.5	19	1.25
Gaming Experience (1-5)	4.23	4	0.81
BTD6 Experience (1-5)	2.46	2	1.47
Hollow Knight Experience (1-5)	2.31	2	1.38
Hades Experience (1-5)	1.92	1	1.44
CS:GO Experience (1-5)	1.92	2	1.07

4.2 Experimental Results Overview

This section outlines the metrics gathered and derived from presentmon. Distributions of this data are also provided to give an overview of the data. These distributions are analyzed to interpret what effect each NetEm setting has on the streamed game.

4.2.1 Presentmon Metrics

In addition to demographic and QoE data, various metrics for performance and latency were measured using presentmon. While the jitter was controlled through NetEm as specified by the user, we measured the jitter on the client machine directly. From the raw presentmon data, a script was used to calculate frames per second (fps), the average number of interrupts per second, and the average total interrupt time per second.

As presentmon records data frame-by-frame, it was trivial to calculate the frames per second across a trial by dividing the number of frames recorded by the total time. With no added jitter, the host-client setup was configured to stream video at 60 fps.

Interrupts were identified by finding delays between frames above the threshold of ~33.33 milliseconds, the length of two frames. For each of these interrupts, the interrupt time above the threshold in milliseconds was recorded as the interrupt length. For a given trial, the total number of interrupts and the sum of interrupt time across those interrupts was calculated by a script. The average number of interrupts was then calculated as the total number of interrupts divided by the trial length. The average interrupt length per seconds was calculated as the total interrupt time divided by trial length. The individual lengths of each interrupt were also recorded.

Figure 12 shows these metrics, per trial, bucketed by the NetEm settings used for that trial. The trial settings are shown on the y axis, with the first number indicating the magnitude setting (in ms) used for that trial, and the second number showing the frequency setting used for that trial. The trial labeled “0,0” is the control trial, where no additional network manipulation was used. Each trial setting includes the number of

trials that were done with that setting. Box and whisker plots are used to display the data. Circles denote outliers, green triangles denote the mean, and vertical orange lines denote the median.

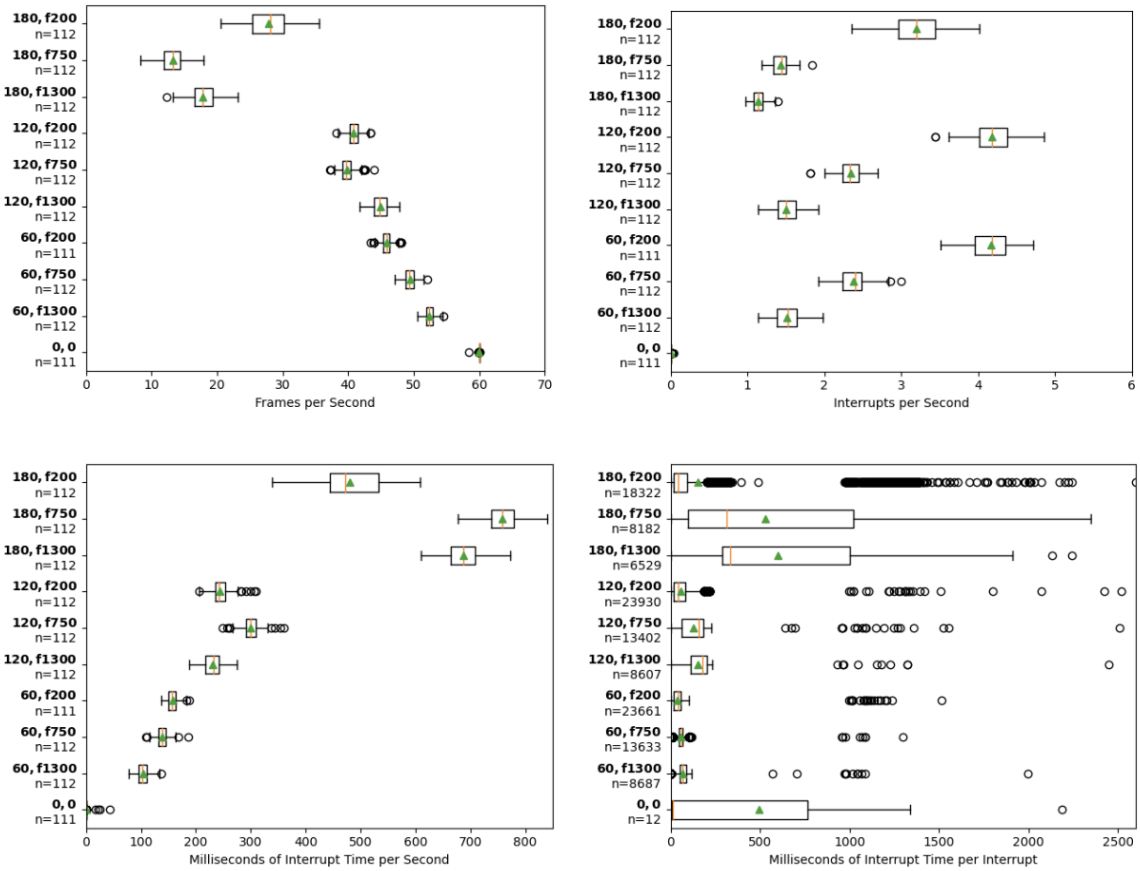


Figure 12: Distribution of various attributes of the presentmon data, per trial type.

The number of frames per second generally trends towards getting smaller with both increasing magnitude and increasing frequency. However, the 120ms/f200 and 180ms/f200 settings have a higher framerate than 120ms/f750 and 180ms/f750 respectively, despite being the greatest two magnitudes and the highest frequency options. We believe that this is due to the frequency being too high to fit the entire large magnitude within a one second interval.

The number of interrupts per second is generally about 1.5 for the f1300 settings, 2.25 for the f750 settings, and 4.0 for the f200 settings. The 180 magnitude versions of these, however, do not follow this pattern quite so closely, and tend to have fewer interrupts per second than their lower magnitude equivalents.

The milliseconds of interrupt time per second generally trends towards getting larger with both increasing magnitude and increasing frequency, however the 120, f200 and 180, f200 settings break this trend and have a smaller amount of interrupt time per second. Since these are the same two settings which have a lower framerate, this reduction in overall interrupt time is likely the reason why those trials have a lower framerate, rather than a major change in the number of interrupts.

The milliseconds of interrupt time per interrupt gives a look into how the frame interrupts appear on the client based on the NetEm settings. The control trial only has 12 interrupts across all 111 control trials, as no interrupts beyond those which would normally occur were present in the control trials. This is an average of 0.1 interrupts per control trial, or one interrupt in every ten control trials. For all non-control trials, there are generally a large number of interrupts lasting some amount less than 500ms, and a noticeable number of outliers starting at about 1000ms, with few outliers between 500ms and 1000ms. The 180 magnitude settings have a smaller total number of interrupts than their 120 magnitude equivalents, however significantly more of their interrupts last longer than 1000ms.

4.2.2 Quality of Experience

In order to analyze the initial effects of each setting on the reported quality of experience, the QoE value reported for each trial was bucketed per setting and plotted as well. Figure 13 shows these distributions.

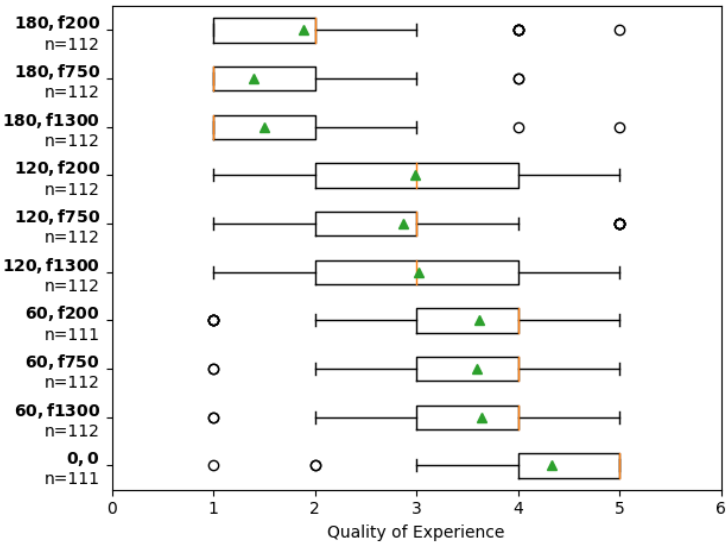


Figure 13: Quality of Experience distributions, per trial type.

From this plot, the most significant differences occur when changing the magnitude setting. Only minor changes happen when changing the frequency setting. With increasing magnitude, the QoE tends to decrease. The control setting had a median of 5, the 60ms magnitude trials had a median of 4, the 120ms magnitude trials had a median of 3, and the 180ms magnitude trials had medians of 1 and 2.

4.3 Data Cleaning

This section outlines the methods and rationale used to clean the data. The overall aim of data cleaning in this study was to remove data that did not accurately reflect participant quality of experience and data that would significantly skew calculations vital to the analysis.

4.3.1 Cleaning based on Experiment Errors

Overall, the experiment proceeded with few major errors. The main issue that arose during the experiment, described in the previous section, was participants misunderstanding game instructions and getting stuck. While care was taken to prevent this after it was initially observed, the trials were not thought to be significantly affected

by this issue. Even if a participant got stuck, it would not result in outliers in the presentmon data. Additionally, the participant's gaming experience did not seem to be significantly impacted based on QoE ratings.

The only instance of data cleaning where we removed data was for two trials where the participant accidentally exited the streaming client. These trials were excluded because they significantly skewed the presentmon metrics. When the participant exited the streaming client, presentmon logged it as a single, extremely long interrupt. The two trials that exhibited this problem were both from the same participant.

4.3.2 Cleaning by QoE Rating

Before proceeding with analysis of participant QoE data, it was necessary to ensure that the QoE data was of good quality. This depended on the quality of participant responses; participants who gave little thought to their responses may give inaccurate QoE ratings. We looked for patterns in the data from individual users that may reflect unreasonable responses. Three main patterns of poor quality data were considered: unconditionally high ratings, unconditionally poor ratings, and no change in ratings across games.

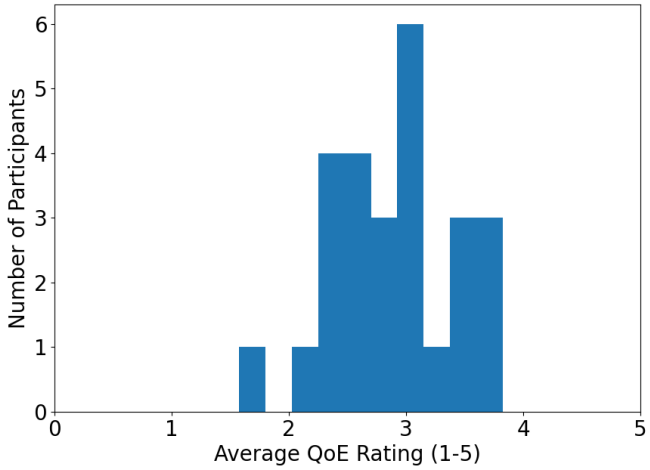


Figure 14: Distribution of mean QoE ratings across all trials per participant.

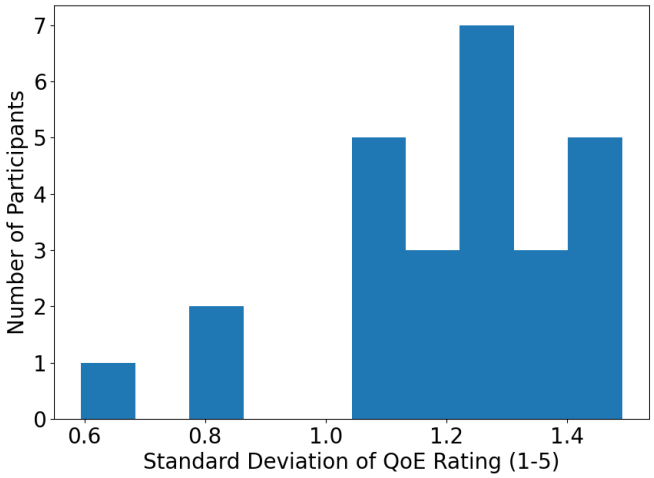


Figure 15: Distribution of standard deviation of QoE ratings across all trials per participant

As shown in Figure 14 above, the majority of participants had QoE rating averages between 2 and 4, which was considered a normal range. Only one participant was identified who gave QoE ratings averaging outside of this range, participant 26 with an average QoE rating of 1.58. As this QoE average is quite low,we scrutinized the QoE ratings of this participant. If the user's pattern of QoE scores was a seemingly random mix of 1's and 2's, regardless of the experimental condition, it may show that the ratings were given carelessly and should not be considered. However, the participant's ratings showed significant variety, with some trials having ratings of 4 or 5.

The standard deviation of participant QoE ratings was also calculated, the distribution of which is shown in Figure 15. Most participants had a standard deviation between 1 and 1.5. 3 participants were found to have a significantly lower standard deviation (< 1). Participants 6, 21, and 23 had QoE rating standard deviations of 0.84, 0.59, and 0.78 respectively. However, all three of these participants had normal average QoE ratings, between 2 and 4, with non-random patterns in their ratings. Based on this analysis, we did not discard any QoE data.

4.4 Significance Testing

Preliminary statistical analysis was done to identify significant differences between experimental conditions as well as confounding factors which may obscure significant trends in the data.

4.4.1 Performance Differences by Game

One concern brought up in preliminary analysis was that different games may perform differently with the same NetEm settings. Therefore, frame per second, average interrupt length per second, and average number of interrupts per second for different trials were compared using an ANOVA test, grouped by game. Table 4 reports the results of these ANOVA tests. No significant difference in any performance metric was observed between trials for different games. This suggests that performance differences between games were not significant and are not a confounding factor.

Table 4: ANOVA test results for comparison of presentmon data between different games

Comparison	F-statistic	p-value
By Game (fps)	0.0149	0.998
By Game (Avg. Interrupt Length)	1.00	0.391
By Game (# of Interrupts/sec)	0.433	0.730

4.4.2 Differences in QoE by Game

One potentially important factor in this experiment was the choice of game, motivated by the idea that differences in gameplay would impact QoE. Table 5 below

shows the results of a one-way analysis of variance (ANOVA) test comparing QoE ratings from trials grouped by game. The result shows that the choice of game does seem to significantly impact QoE.

Table 5: ANOVA test results for comparison of QoE data between different game trials

Comparison	F-statistic	p-value
By Game	4.01	7.53e–3

To better understand this effect, pairwise t-tests between each group of trials were also performed. The results are shown in Table 6 below. The main point of interest is the statistical difference ($p < 0.05$) shown between QoE ratings for Bloons Tower Defense 6 and those for all other games. However, no significant difference is observed between any of the other three games. Additionally, with Bonferroni correction applied, only Bloons Tower Defense 6 and CS:GO are statistically different ($p < 0.0083$).

Table 6: Pairwise t-tests between QoE ratings for trials grouped by game

Game 1	Game 2	p-value	t-statistic
Bloon Tower Defense 6	Hollow Knight	0.012	-2.52
	Hades	0.012	-2.53
	CS:GO	0.0017	-3.15
Hollow Knight	Hades	1.0	0.00
	CS:GO	0.56	-0.590
Hades	CS:GO	0.56	-0.590

As shown in Figure 16 below, the mean QoE rating for Bloons Tower Defense 6 is higher than those of other games. However, the difference is relatively small, amounting to not even half a point on the 1-5 scale.

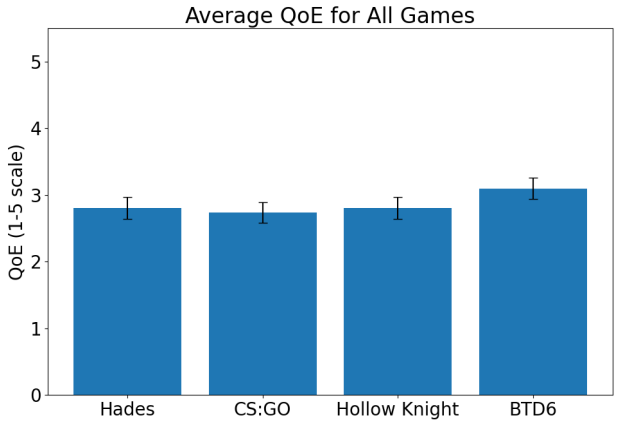


Figure 16: Average QoE ratings for each game with error bars representing 95% confidence intervals.

4.4.3 Differences in QoE by Settings

After confirming consistent performance metrics and generally consistent QoE ratings between games, the next step was to identify differences between experimental conditions. As a preliminary step, ANOVA tests were performed comparing QoE ratings between different settings. Table 7 below reports the results of these tests comparing QoE ratings for trials with different magnitude and frequency settings. While the magnitude setting was shown to be significant, no significant difference was found in QoE ratings between frequency conditions. This matches the general trend in means QoE ratings for each condition, shown in Figure 17. While the ANOVA test does not necessarily confirm the observed downward trend in QoE at higher magnitudes, it does suggest that significant differences do exist. The result for frequency settings also matches the observed means for different frequencies and the lack of difference between them.

Table 7: ANOVA test results for comparison of QoE values between trials with different jitter settings (magnitude/frequency)

Comparison	F-statistic	p-value
Magnitude Settings (60ms, 120ms, 180ms)	379.47	2.55e-121
Frequency Settings (f200, f750, f1300)	1.9527	0.142

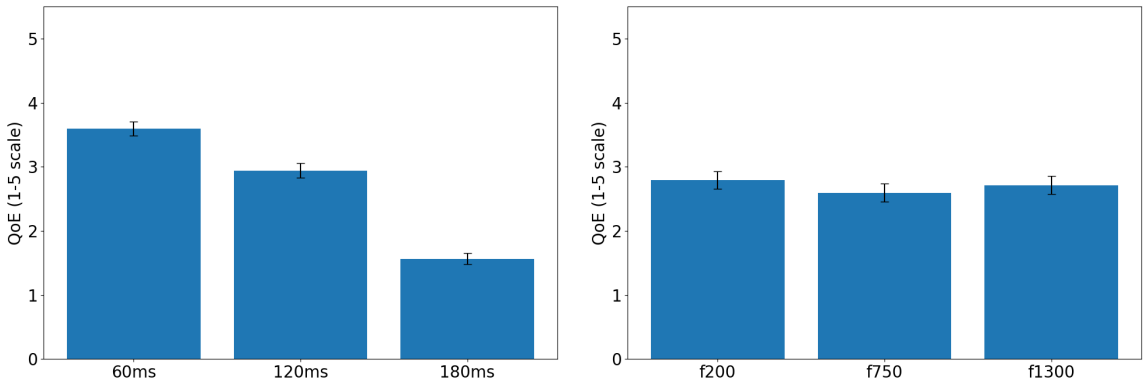


Figure 17: Barplots showing average QoE values at different magnitude (left) and frequency (right) settings for jitter with 95% confidence intervals.

While the initial ANOVA test showed no significant difference between frequency settings, it was suggested that difference in magnitude may affect the data in such a way that significant differences between frequency settings are obscured. To test this, ANOVA tests were performed comparing QoE values between trials with different frequency settings but the same magnitude settings (e.g., compare all trials with the 60ms magnitude condition grouping by frequency setting). The results of these tests are reported in Table 8 below. At the 60ms and 120ms magnitude settings, the difference in QoE ratings between trials with different frequency settings was still insignificant. However, a significant difference was observed at the 180ms setting. The reasons for this difference specifically at the 180ms setting are unclear, but potential explanations

are discussed later in this report. Overall, the results of these tests support the conclusions that frequency does not significantly affect QoE.

Table 8: ANOVA test results for QoE values between trials with different jitter frequency settings for individual magnitude conditions

Comparison	F-statistic	p-value
Frequency Settings (60ms jitter magnitude)	0.151	0.860
Frequency Settings (120ms jitter magnitude)	0.584	0.558
Frequency Settings (180ms jitter magnitude)	11.6	1.38e-5

Additionally, a two-way ANOVA test run on the average and standard deviation of each combination of NetEm settings and game tested shows the extent to which the frequency and magnitude affect QoE as well as the extent to which their effects are influenced by the game being tested. These results have some implications on models for QoE, but also illustrate a need to normalize the QoE data for each game played, which is explored in the next section. The results of these tests are shown in Table 9.

Table 9: P-values found with two-way ANOVAs for QoE scores bucketed by settings the extent to which each settings' scores are affected by the game being tested. Color-coded such that p<.01 is green, p>.05 is red, and others are yellow.

Comparison	Affects mean	Mean affected by game	Affects std deviation	Std deviation affected by game
Magnitude settings	<0.0001	<0.0001	0.0335	0.613
Frequency settings	0.048	0.173	0.0158	0.250
All NetEm settings	<0.0001	0.0104	<0.0001	0.0829

4.4.4 Difference in QoE Z-scores by Settings

In addition to QoE analysis, the data was normalized by taking the z-score of each trial relative to the user's other QoE scores for the game they were playing. This allowed for analysis irrespective of the differences in QoE between users and between each game, meaning it can be used as a metric for how much the changing NetEm settings affected the user's experience. With the z-scores grouped into each of the ten combinations of frequency and magnitude settings, an ANOVA test revealed that these settings significantly influenced the z-scores. Two further ANOVA tests showed that the three non-control NetEm settings for magnitude and frequency influenced the QoE z-scores irrespective of one another. Due to our setup also resulting in the average frame rate of a trial correlating highly with the average interrupt magnitude, the significance of frame rate was also tested by taking the mean framerate between each of the magnitude settings and categorizing trials by which of those its average frame rate was closest to. The results of these tests are shown in Table 10 below.

Table 10: ANOVA test results for QoE z-score values between different bucketings of NetEm settings

Comparison	F-statistic	p-value	F-crit
All NetEm settings	210.91	0	1.8890
Magnitude settings (control excluded)	701.31	0	3.0054
Frequency settings (control excluded)	3.3893	0.0341	3.0054
Framerate groups	608.61	0	3.0054

Moreover, these tests were done with each of the bucketing categories and the game being tested to guide the modeling process. A two-way ANOVA was run on the means and standard deviations of each of the combinations of buckets and games to determine which variables are suitable for the purpose of modeling. The p-values for each of the buckets are presented here in Table 11 so as to make the data more readable, as in all cases when the p-value was below 0.05 the F-statistic was greater than the F-critical value. These results validate the assumption that this normalizing process gives an approximation of QoE which is not significantly affected by the game being played. More detailed implications of this data are discussed later in this report.

Table 11: P-values found with two-way ANOVAs for QoE z-score values grouped by settings and by framerate and the extent to which each category of buckets' z-scores are affected by the game being tested. Color-coded such that p<0.01 is green, p>0.05 is red, and others are yellow.

Comparison	Affects mean	Mean affected by game	Affects std deviation	Std deviation affected by game
Magnitude settings	<0.0001	0.934	0.551	0.340
Frequency settings	0.0300	0.963	0.0234	0.874
Framerate buckets	<0.0001	0.758	0.00899	0.571
All buckets	<0.0001	0.992	<0.0001	0.0951

4.5 Trends in Data

This section outlines the statistical analysis performed with a focus on identifying and characterizing significant relationships in the data.

4.5.1 Univariate Regression (QoE)

Ordinary least-squares regression was performed using each of the three presentmon metrics to fit separate models for QoE. The results of these regressions are summarized in Table 12 below. Both frames per second and average interrupt length per second were found to be significant predictors of QoE. The number of interrupts per second was not found to be significant from this regression. All regressions show poor fitting on the data. Plots of experimental data with regressions lines are shown in Figure 18 below.

Table 12: Univariate regression summary

Predictive Variable	Coeff.	p-value	Intercept	R^2
avg. interrupt length/sec	-0.004	0.00	4.05	0.49
# of interrupts/sec	-0.039	0.22	2.95	0.001
frames/sec	0.065	0.00	0.335	0.49

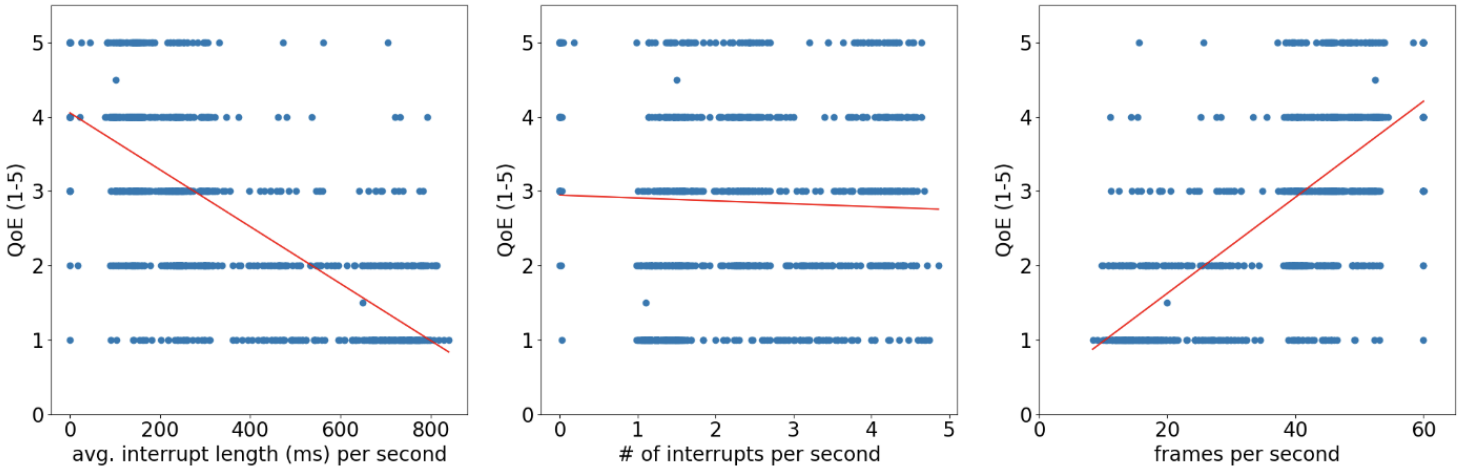


Figure 18: Regression lines for average interrupt length/sec (left), interrupts/sec (center), and fps (right) with scatterplots of experimental data

Following the initial regressions on the raw data, it was suggested that performing regressions with average QoE as the response variable may better illustrate relationships in the data. To do this, kernel density estimation was used to estimate the distribution of QoE ratings across the range of values for each presentmon metric. This was possible because participants in almost all cases chose to give integer ratings, resulting in a discrete scale of QoE. Two trials did have decimal values for QoE rating and were excluded from this analysis.

Kernel density estimation was performed for each QoE rating 1-5 for each presentmon metric. These estimates were then corrected so as to reflect the proportions of QoE ratings observed in the data. These estimated distributions are shown as curves in Figure 19 below.

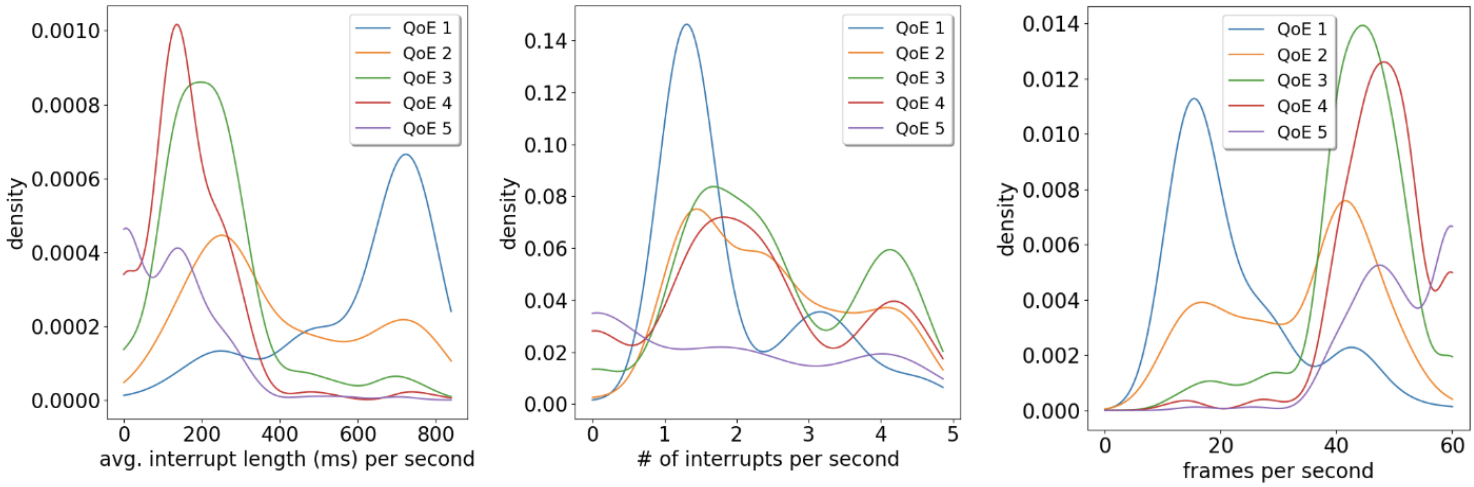


Figure 19: Distributions of QoE ratings 1-5 across ranges of average interrupt length/sec (left), interrupts/sec (center), and fps (right)

These curves were then used to calculate average QoE estimates. Average QoE was estimated as a weighted average of the densities of QoE ratings for a given presentmon metric value. Using this method, it was possible to provide estimates for average QoE across the full range of a given presentmon metric. Regressions were once again performed using the QoE estimates as a response variable. The results of these regressions are shown in Table 13 below. Once again, average interrupt length per second and frames per second were found to be significant predictors of the estimated average QoE, while the number of interrupts per second was not. These regressions also show better fit as compared to the regression on the raw data, wth $R^2 >$ 0.9 for average interrupt length per second and fps. Graphs showing a curve of the average QoE estimate as well as the regression line using that estimate are shown below in Figure 20.

Table 13: Univariate regressions with average QoE summary

Predictive Variable	Coeff.	p-value	Intercept	R^2
avg. interrupt length/sec	-0.004	0.00	3.92	0.943
# of interrupts/sec	-0.033	0.142	2.99	0.011
frames/sec	0.062	0.00	0.3928	0.957

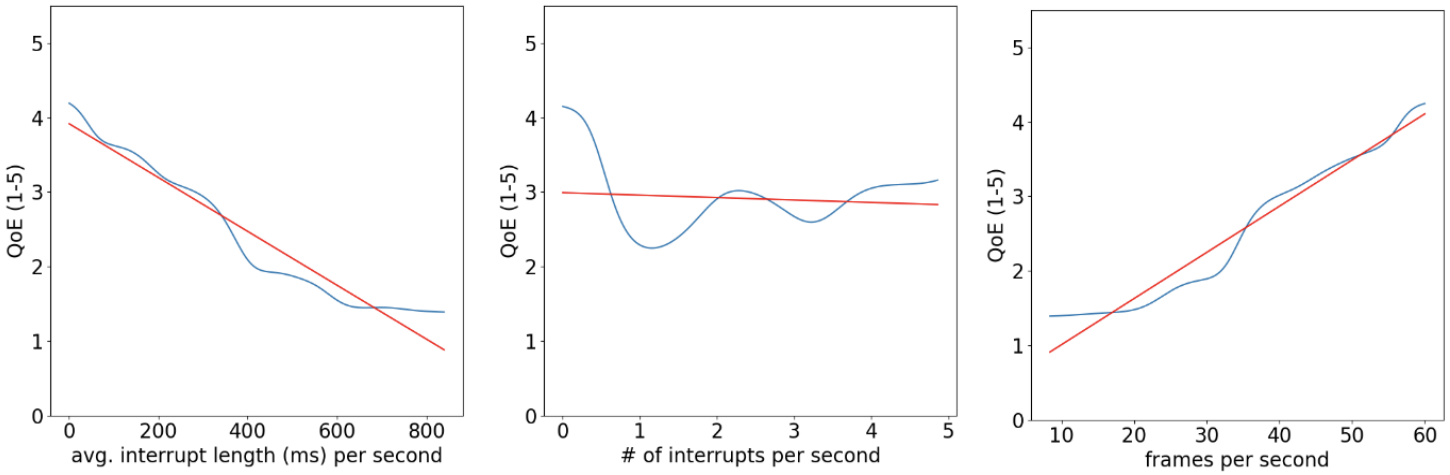


Figure 20: Plots of estimated avg. QoE curves and regression lines using average interrupt length/sec (left), interrupts/sec (center), and fps (right)

4.5.2 Univariate Regressions (Z-Score)

Least-squares regression was performed using the three presentmon metrics for separate models of the z-score normalizations of QoE. The results showed that similar to QoE, the average interrupt length per second and frames per second were significantly correlated with z-score while the number of interrupts per second was less

predictive. Also similarly to the QoE regressions, all regressions were poor fits for the data. The regression results are shown in Table 14 below.

Table 14: Univariate regression summary (z-score)

Predictive Variable	Coeff.	Intercept	R^2
avg. interrupt length/sec	-0.003	1.01	0.62
# of interrupts/sec	-0.0235	0.0531	0.001
frames/sec	0.0549	-2.15	0.62

Further analysis used k-means clustering to sort each non-control trial into the closest of three buckets for each metric. The cluster centers are reported in Table 15 below.

Table 15: K-mean cluster centers

Presentmon Metric	Low cluster center	Medium cluster center	High cluster center
avg. interrupt length/sec	195.2	474.2	722.6
# of interrupts/sec	1.4	2.5	4.1
frames/sec	17.3	38.3	48.2

Of the 27 possible combinations of these buckets, each non-control trial fit into one of 14 combinations, three of which fit exactly one trial. An ANOVA test was run on these 14 combinations as buckets for the z-scores which determined they were a significant grouping (with a p-value of 0 and an F-statistic of 100.0 against an F-critical value of 1.73). Least-squares regression was then run on the mean and standard

deviation (where applicable) of each of these combinations using each presentmon metric. The results are shown in Table 16 below.

Table 16: Univariate regression for mean and standard deviation summary

Value predicted	Predictive Variable	Coeff.	Intercept	R^2
Mean	avg. interrupt length/sec	-0.002	0.568	0.444
	# of interrupts/sec	0.0243	-0.371	0.001
	frames/sec	0.0438	-1.74	0.63
Standard deviation	avg. interrupt length/sec	-3.3e-4	0.675	0.153
	# of interrupts/sec	-0.0385	0.668	0.076
	frames/sec	8.69e-3	0.252	0.466

These regressions in conjunction allow for the calculation of a 95% confidence interval given one of the presentmon metrics, although the reliability of these predictions appears to correspond with the p-values found previously. Graphs showing these intervals superimposed over the z-score plots themselves can be seen in Figure 21.

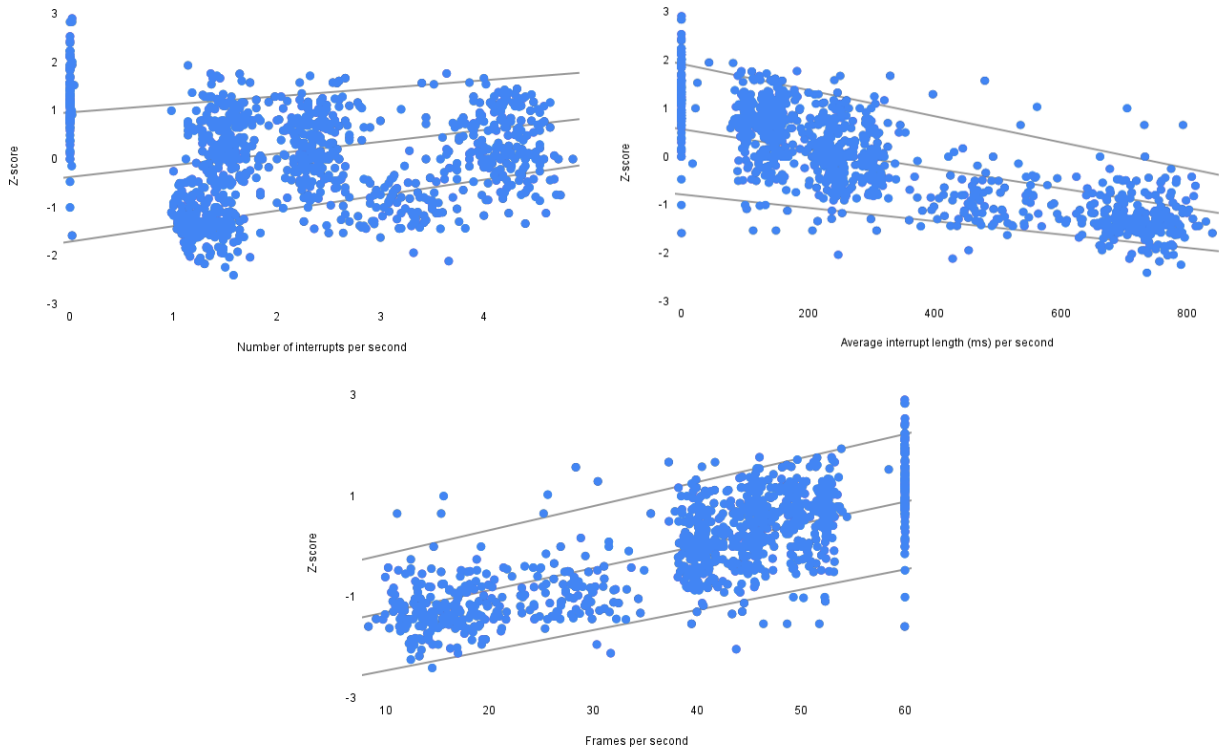


Figure 21: Plots of z-score for number of interrupts per second (left), average interrupt length (right), and frames per second (bottom) alongside their predicted means and 95% confidence intervals

4.5.3 Multivariate Regressions (QoE)

In addition to univariate regressions, multivariate regressions were performed with average interrupt length per second and number of interrupts per second as predictors. While average interrupt length per second was the only one of the two found to be significant in univariate analysis, multivariate regression using both variables was considered to provide an alternate perspective on these relations.

Two regressions were performed, with one fitted on all of the data and the other fitted on data excluding 180ms magnitude and f200 frequency conditions. This was done because high magnitude or frequency settings may have limited the number of interrupts possible per second and vice versa. Therefore, at high magnitude or frequency conditions, there may be some negative covariance between the average interrupt length per second and the number of interrupts per second.

The results of multivariate regressions using average interrupt length per second and interrupts per second are shown below in Table 17. The regression fit on the complete dataset showed both the average interrupt length per second and the number of interrupts per second to be significant predictors of QoE. The regression fit on data excluding 180ms magnitude and f200 frequency conditions showed only average interrupt length per second to be significant. While the first regression seems to conflict with univariate regressions showing the number of interrupts per second to be insignificant in predicting QoE, the results of the second regression aligns with the univariate regression results. This may be due to the covariance effect previously mentioned. As such, average interrupt length per second remains the only statistically significant predictor.

Table 17: Multivariate regression results

Data Used	R^2	Predictor	Coeff.	p-value
All data	0.493	[Constant]	4.23	0.00
		Avg. interrupt length/sec	-0.0038	0.00
		Interrupts/sec	-0.080	0.00
180ms, f200 excluded	0.529	[Constant]	4.26	0.00
		Avg. interrupt length/sec	-0.0042	0.00
		Interrupts/sec	-0.076	0.29

As with the univariate regressions, further multivariate analysis focused on interpreting estimates of average QoE data. Multivariate kernel density estimates were used to find average QoE estimates varying by the average interrupt length per second and the number of interrupts per second. Plots showing the distribution of QoE ratings density estimates are shown below in Figure 22.

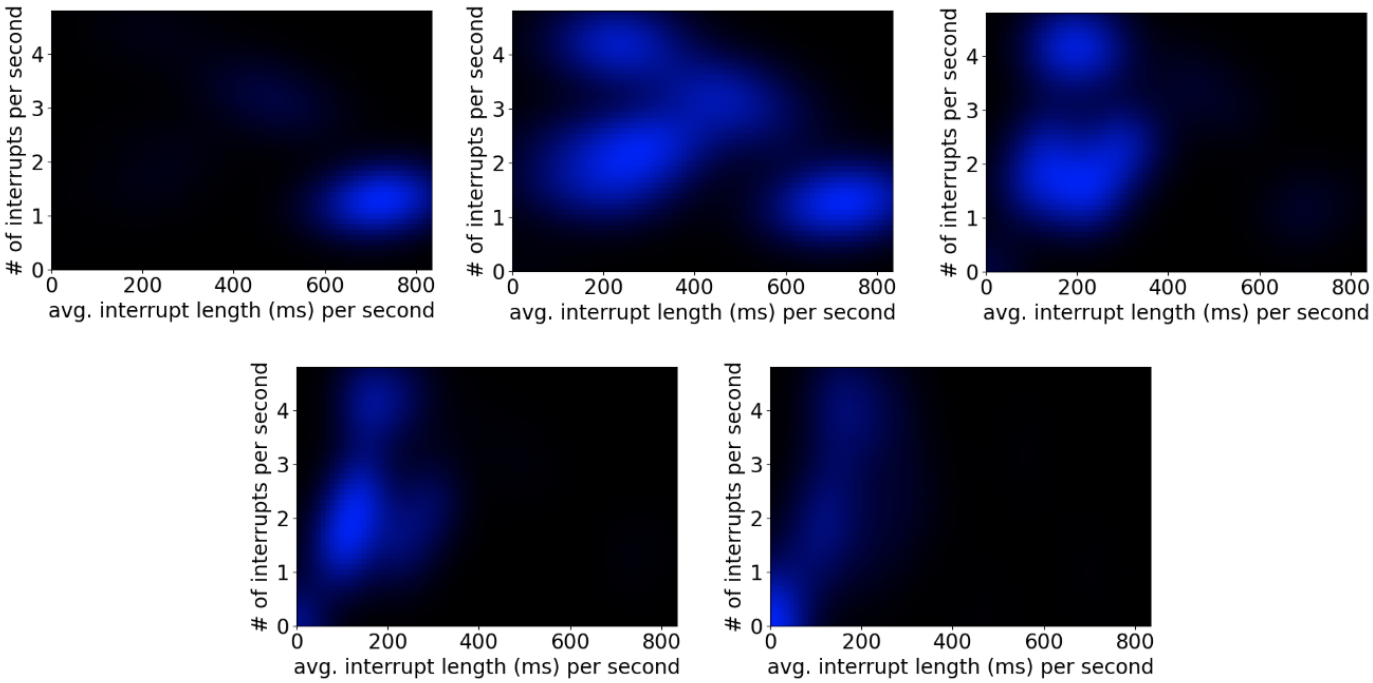


Figure 22: Density plots for each QoE rating 1-5 (ordered left to right, up to down). Black to blue color gradient represents increasing density.

Following the same approach as the univariate density estimate, the average QoE ratings across this space were determined by a weighted average of the densities for each QoE rating. The average QoE across varying average interrupt length per second and number of interrupts per second is shown below in Figure 23.

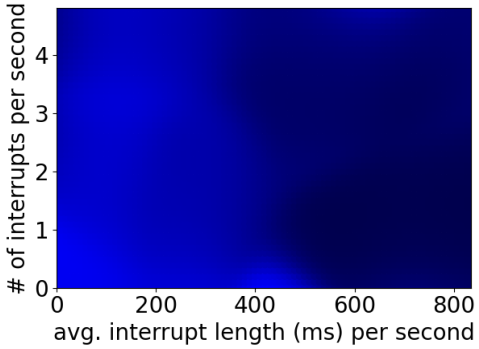


Figure 23: Density plot for average QoE. Black to blue color gradient represents increasing QoE (1-5)

These plots visually confirm an inverse relationship between average interrupt length per second and average QoE. Additionally, the plot for average QoE seems to show a weak trend between the number of interrupts per second and average QoE, in contrast to the regression results. However, it is important to note that average QoE estimation across multiple variables involves much more extrapolation, reducing the accuracy of the estimates. Figure 24 below shows the spread of trials across the same space, showing that coverage may be insufficient to estimate QoE across both axes with significant confidence.

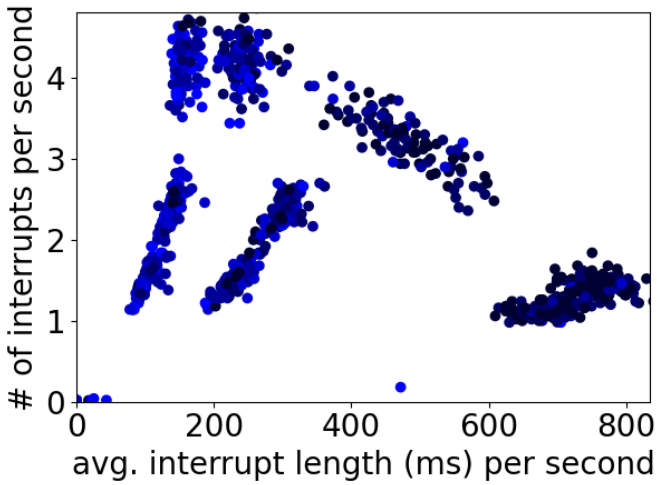


Figure 24: Scatter plot of individual QoE ratings with avg. interrupt length/sec and interrupts/sec as x and y axes respectively. Black to blue color gradient represents increasing QoE.

5. Conclusion

This study aimed to build on previous work in the area of latency in cloud gaming systems with a focus on understanding how user experience is impacted by latency. Of specific interest to this study is work investigating jitter, a type of non-constant latency. For cloud-based game streaming, jitter is characterized by the frequency and magnitude of interrupts to the delay of video frames, which are delay spikes in a connection.

The specific goal of this study was to better characterize the relationship between jitter and user experience. While previous work has established the effects of delay on user experience, the topic of jitter in cloud gaming systems is still relatively unexplored. Establishing the effects of jitter frequency and magnitude on user experience would provide new insights. This study was designed to accomplish this by simulating jitter conditions on a controlled setup and measuring user experience at different conditions.

Results from this study successfully identified significant effects of jitter on user experience. A strong relationship was found between jitter magnitude and user QoE, showing that increases in jitter magnitude generally lead to declines in user experience. The study also characterized the relationship between jitter frequency and QoE, though these results were less conclusive. The frequency of interrupts did not have a large effect on user QoE, though some evaluations suggested that more interrupts did lead to worse user experience.

Additionally, the results of this study shed light on how user experience in different games is impacted by jitter. While all games tested in this study were similarly affected, the baseline QoE values suggest that users may find some gameplay experiences better or worse than others regardless of jitter. However, it was confirmed that declines in user experience are similar across different games. Analysis was also conducted using z-scores to normalize QoE values and remove differences in QoE by game or participant bias. This analysis matched conclusions drawn from non-normalized QoE analysis, showing that these trends occur regardless of game choice or user preference.

Lastly, the results of this study further established relationships between jitter and QoE through predictive models. Models predicting QoE using jitter magnitude showed an approximately linear relationship between the two, suggesting that equal increases in

magnitude degrades user experience regardless of the initial and final magnitude. Models using jitter frequency as a predictor did not establish a significant relationship with user QoE. However, models including both magnitude and frequency as predictors did identify both as being significant predictors of QoE, suggesting some effect of jitter frequency on QoE. As with the results mentioned above, z-score based analysis confirmed that game choice and user preference did not cause these differences.

6. Future Work

Future work could further investigate the effect of jitter magnitude on QoE. While magnitude appears to be a significant predictor of QoE, the exact relationship could be more thoroughly understood. A linear model as used in this study provided a decent fit, but a validation of this model has not been performed. Future studies could aim to verify the linear model for magnitude versus QoE by gathering a larger data set on which to train and test a model. Non-linear models could also be explored and assessed. If successful, this could provide a more accurate model for average QoE useful for other research.

Additionally, the methods of QoE measurement used in this study could be further developed. The kernel density estimates provided a generally decent estimate of average QoE for the univariate regressions, but did not seem to hold up as well in the multivariate context. Using a wider range of experimental conditions to give better coverage of the range of data may improve the performance of this method. Determining average QoE in a purely experimental way may also be useful, if only in assessing the accuracy of the KDE-based estimate. These steps would allow for more accurate and reliable measurement of average QoE that would aid in assessing user experience in other studies.

Future studies could also expand on QoE measurement by making it easier for participants to give decimal values. While this study aimed to do so, participants almost always gave integer values for QoE. Having a wider range of possible values could reveal clearer trends in the data. Other user experience metrics could be included as well, such as user performance. Expanding on previous metrics and including others would provide a more well-rounded assessment of user performance which may reveal effects not seen from a more limited perspective.

The attempts to find differences in camera type did not result in significant differences between the games studied. However, the game selection in this study was limited and faced issues with confounding variables of player skill and image complexity. Future studies could attempt to account for these variables in game selection and in data analysis. This would allow for more confident, general conclusions about games as a whole.

Further work could also be done to investigate the effects of interrupt frequency. The results from this study generally showed that frequency was not significant, but many factors prevented a thorough analysis. Experimental design meant to specifically investigate frequency may help clarify these results. This could be done by reconfiguring the NetEm settings to allow for more control of the number of magnitude of interrupts. This would allow for more confident assessments of the true effect of frequency on user experience.

References

J. Allard, A. Roskuski and M. Claypool, “Measuring and Modeling the Impact of Buffering and Interrupts on Streaming Video Quality of Experience”, In Proceedings of the 18th International Conference on Advances in Mobile Computing & Multimedia (MoMM), Chiang Mai, Thailand, 30 November - 2 December 2020. Online: http://www.cs.wpi.edu/~claypool/papers/buff-int/

D. Austerberry, “The Technology of Video and Audio Streaming”. 2nd ed., Focal, 2005.

N. Barman, S. Zadtootaghaj, S. Schmidt, M. G. Martini and S. Möller, "GamingVideoSET: A Dataset for Gaming Video Streaming Applications," 16th Annual Workshop on Network and Systems Support for Games (NetGames), 12-15 June 2018, pp. 1-6, doi: 10.1109/NetGames.2018.8463362.

M. Bergeron, Cameron Gutman, Diego Waxemberg, R. Aidan Campbell, “Moonlight Game Streaming Project”, Accessed 3 March 2023, https://github.com/moonlight-stream.

K. Browning, “'Crucial Time' for Cloud Gaming, Which Wants to Change How You Play.” *The New York Times*, The New York Times, 1 July 2021, https://www.nytimes.com/2021/07/01/technology/cloud-gaming-latest-wave.html.

M. Claypool and K. Claypool, “Latency can kill: precision and deadline in online games”, In Proceedings of the first annual ACM SIGMM conference on Multimedia systems (MMSys '10). Association for Computing Machinery, New York, NY, USA, 22 February 2010, 215–222. doi: 10.1145/1730836.1730863

R. Darcey, B. Han, W. Zhang, “Jitter and Latency in Cloud Game Streaming, 2022

J. Gaskin. “Thin vs. Thick Clients.” *Technology Solutions That Drive Business*, 8 September 2022, https://biztechmagazine.com/article/2011/09/thin-vs-thick-clients.

S. Hemminger, “tc-netem”, Accessed 3 March 2023, https://www.linux.org/docs/man8/tc-netem.html.

A. Horé and D. Ziou, "Image Quality Metrics: PSNR vs. SSIM," *20th International Conference on Pattern Recognition*, 23-26 August 2010, pp. 2366-2369, doi: 10.1109/ICPR.2010.579.

Q. Huynh-Thu, M. Ghanbari, “The Accuracy of PSNR in Predicting Video Quality for Different Video Scenes and Frame Rates”, *Telecommun Systems,* 2012, 49, pp. 35–48, doi: 10.1007/s11235-010-9351-x.

International Telecommunications Union, “Subjective Video Quality Assessment Methods for Multimedia Applications”, April 6 2008, E 33690, Online: https://www.itu.int/rec/T-REC-P.910-200804-S/en.

J. Montgomery, “PresentMon”, Accessed 3 March 2023, https://github.com/GameTechDev/PresentMon

S. Naji, and D. Abrahams, “Cloud Gaming Is Being Sold as the next Stage of Video Game Innovation, but Consumers Will Have the Last Word.” *GamesIndustry.biz*, GamesIndustry.biz, 14 March 2022, https://www.gamesindustry.biz/cloud-gaming-is-being-sold-as-the-next-stage-of-video-game-innovation-but-consumers-will-have-the-last-word-opinion.

Open-Stream Team, “Open-Stream”, Accessed 3 March 2023, https://github.com/LS3solutions/openstream-server.

C. Perkins et al., “A Survey of Packet Loss Recovery Techniques for Streaming Audio.” IEEE Network, vol. 12, no. 5, 1998, pp. 40–48, doi: 10.1109/65.730750.

K. A. Rahman, R. McCool, and G. Somadder, “Gaming in the Cloud: A Technical Deep Dive”, Technical Talk at the Game Developer’s Conference (GDC) (presented by Google), San Francisco, CA, USA, March 19, 2019. https://youtu.be/K33gctpveuk.

S. Schmidt, S. Zadtootaghaj, and S. Moller, "Towards the delay sensitivity of games: There is more than genres," in Ninth International Conference on Quality of Multimedia Experience (QoMEX), Erfurt, Germany, May 2017, pp. 1–6. doi: 10.1109/QoMEX.2017.7965676.

K. Shatzkamer, “Google Cloud Brandvoice: Living on the Edge: How next-Gen Mobile Networks Will Drive the Evolution of Cloud Computing.” Forbes, Forbes Magazine, 21 April 2022, www.forbes.com/sites/googlecloud/2021/10/06/living-on-the-edge-how-next-gen-mobile-networks-will-drive-the-evolution-of-cloud-computing/.

A. Wahab, N. Ahmad, M. G. Martini, and J. Schormans, "Subjective Quality Assessment for Cloud Gaming," J, vol. 4, no. 3, pp. 404–419, August 2021, doi: 10.3390/j4030031.

Z. Wang, A. C. Bovik, H. R. Sheikh and E. P. Simoncelli, "Image Quality Assessment: From Error Visibility to Structural Similarity," in IEEE Transactions on Image Processing, vol. 13, no. 4, pp. 600-612, April 2004, doi: 10.1109/TIP.2003.819861.

Appendix

Participant Question Sheet

Background Information

Some questions are not required. Please fill this section out before continuing to the experiment. For questions that ask you to rank from 0 to 5, consider the following scale:

1 - Very unfamiliar

2 - Somewhat unfamiliar

3 - A little familiar

4 - Somewhat familiar

5 - Very familiar

What is your age?

_____

What is your gender?

- ☐ Male
- ☐ Female
- ☐ Other _____

Rate your level of experience with video games in general from 1 to 5 (required)

_____

Rate your level of experience with CS:GO from 1 to 5 (required)

_____

Rate your level of experience with Hades from 1 to 5 (required)

_____

Rate your level of experience with Hollow Knight from 1 to 5 (required)

_____

Rate your level of experience with Bloons Tower Defense 6 from 1 to 5 (required)

_____

Game 1

Please follow this rating scale:

1 - Bad

2 - Below Average

3 - Average

4 - Above Average

5 - Good

1. Rate the quality of experience of the last segment of gameplay from 1 to 5

_____

2. Rate the quality of experience of the last segment of gameplay from 1 to 5

_____

(Each game had 10 total locations to rate QoE, with 4 total games).

Router Machine Script

```
import random, os
import numpy as np
import pandas as pd
from datetime import datetime, timedelta
import time


games = ["Hades", "Hollow Knight", "Bloons TD6", "CS:GO"]

commands = ["echo",
            "sudo tc qdisc add dev eth0 root netem delay 0.1ms 60ms distribution f200 rate 1000mbit",
            "sudo tc qdisc add dev eth0 root netem delay 0.1ms 120ms distribution f200 rate 1000mbit",
            "sudo tc qdisc add dev eth0 root netem delay 0.1ms 180ms distribution f200 rate 1000mbit",
            "sudo tc qdisc add dev eth0 root netem delay 0.1ms 60ms distribution f750 rate 1000mbit",
            "sudo tc qdisc add dev eth0 root netem delay 0.1ms 120ms distribution f750 rate 1000mbit",
            "sudo tc qdisc add dev eth0 root netem delay 0.1ms 180ms distribution f750 rate 1000mbit",
            "sudo tc qdisc add dev eth0 root netem delay 0.1ms 60ms distribution f1300 rate 1000mbit",
            "sudo tc qdisc add dev eth0 root netem delay 0.1ms 120ms distribution f1300 rate 1000mbit",
            "sudo tc qdisc add dev eth0 root netem delay 0.1ms 180ms distribution f1300 rate 1000mbit"]

commandsValues = {"echo":[0, 0],
            "sudo tc qdisc add dev eth0 root netem delay 0.1ms 60ms distribution f200 rate 1000mbit":["60",
"f200"],
            "sudo tc qdisc add dev eth0 root netem delay 0.1ms 120ms distribution f200 rate 1000mbit":["120",
"f200"],
            "sudo tc qdisc add dev eth0 root netem delay 0.1ms 180ms distribution f200 rate 1000mbit":["180",
"f200"],
            "sudo tc qdisc add dev eth0 root netem delay 0.1ms 60ms distribution f750 rate 1000mbit":["60",
"f750"],
            "sudo tc qdisc add dev eth0 root netem delay 0.1ms 120ms distribution f750 rate 1000mbit":["120",
"f750"],
            "sudo tc qdisc add dev eth0 root netem delay 0.1ms 180ms distribution f750 rate 1000mbit":["180",
"f750"],
            "sudo tc qdisc add dev eth0 root netem delay 0.1ms 60ms distribution f1300 rate 1000mbit":["60",
"f1300"],
            "sudo tc qdisc add dev eth0 root netem delay 0.1ms 120ms distribution f1300 rate 1000mbit":["120",
"f1300"],
            "sudo tc qdisc add dev eth0 root netem delay 0.1ms 180ms distribution f1300 rate 1000mbit":["180",
"f1300"]}

subjectID = input("subjectID: ")

random.shuffle(games)

df = pd.DataFrame(columns=["game", "mag", "freq", "start", "end"])

for game in games:
    os.system("sudo tc qdisc del dev eth0 root")
    print("Set up " + game + " for the participant")
    print("Allow user to practice the game for a couple of minutes")
    random.shuffle(commands)
    for idx, command in enumerate(commands):
        input("TRIAL " + str(idx + 1) + " - press enter to start the trial")
        start = datetime.today()
        os.system("sudo tc qdisc del dev eth0 root")
        os.system(command)
        time.sleep(50)
        print("STOP! Tell the user to stop playing now!")
        df.loc[len(df.index)] = [game, commandsValues[command][0], commandsValues[command][1], start,
start+timedelta(seconds=50)]
os.system("sudo tc qdisc del dev eth0 root")
df.to_csv(subjectID + "results.csv")
```

Client Machine Script

```
import sys,os,time
import ctypes

def is_admin():
    try:
        return ctypes.windll.shell32.IsUserAnAdmin()
    except Exception as e:
        print(e)
        return False

if not is_admin():
    print("Please run this file with right click + 'run as administrator'.")
    input()
    sys.exit(0)

identifier = input("Enter the participant identifier: ")
stamp = time.strftime('%y.%m.%d-%H.%M.%S')


fileName = identifier+'_'+stamp+'.csv'

os.system('C:/Users/claypool/Desktop/PresentMon-1.8.0-x64.exe -process_name Moonlight.exe -output_file
C:/Users/claypool/Desktop/STADIA_IQP/results/%s  -no_top'%(fileName))
```

Recruiting Email

Hello everyone!

As part of our IQP, we are looking for participants to take part in a user study that explores how latency and jitter can impact the user's experience with cloud-based game streaming.

The study should take about an hour per person, and involve playing four different games and answering survey questions as you play.

You will be compensated with a $15 Amazon Gift Card for your time.

Additionally, participation can count for IMGD playtesting credit, if you need it.

Participation is voluntary, you may leave at any time. The study does include a game with many bright flashing lights, so if you are prone to epileptic seizures please do not participate. Apart from this, there is no/minimal risk for this user study and this study has been approved by the IRB at WPI.

To sign up for a slot, please pick a time on slottr [omitted].

If you have any further questions, feel free to reach out to any of us:

Thomas Flanagan (tmflanagan@wpi.edu)

Carter Nakagawa (clnakagawa@wpi.edu)

Michael Oliveira (mjoliveira@wpi.edu)

Professor Claypool (claypool@wpi.edu)

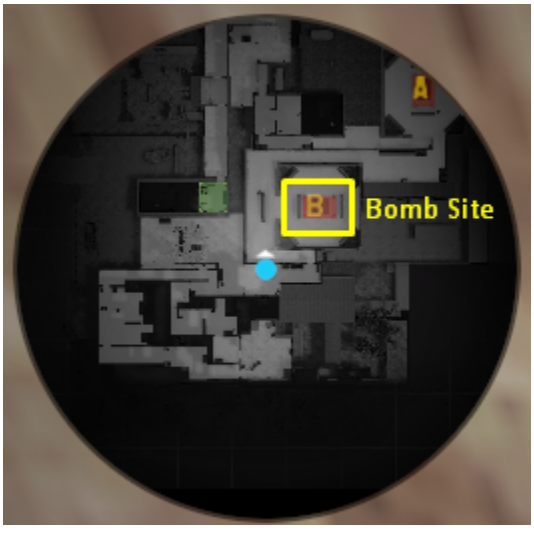
Game Tutorial Sheets

CS:GO

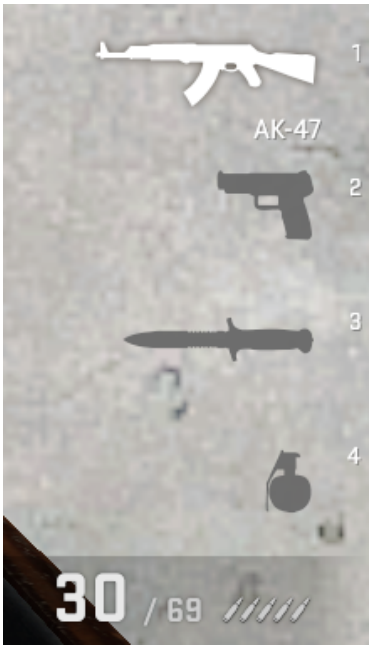
Mouse - Moves your camera, allowing you to look in a different direction.
Left Mouse Button - Fires your weapon. Some weapons (like pistols) must be tap fired by repeatedly pressing the button, some (like SMGs, rifles) allow you to hold down the button to spray or tap the button to fire slower but more accurately.
Scroll wheel - Cycle between weapons.
W A S D - Moves your position in space, used similar to arrow keys.
E - Picks up or put down a weapon where you are looking at. Also used to open doors.
R - Reloads your weapon.
Ctrl - Hold to “kneel”/”crouch”, lowering your height and making you move slower.
Space bar - Jump. Can be used to get over short obstacles.
Esc - Pause the game.

Firing individual shots or in short bursts is more accurate than holding down the fire button.

Your radar is at the top left of your screen. It shows the layout of the map and the location of bomb sites. Bomb sites are labeled with a capital letter and a red coloring.



Your weapons and ammo count are at the bottom right of your screen. You can switch to a different weapon using the number key next to a weapon, or by scrolling to it with the scroll wheel.



Hades

W - Move forward
A - Move left
S - Move backward
D - Move right
Space - Dash (can be combined with W A S D to specify direction)
Left Click - Normal attack (can be combined with dash for a dash-attack)
Right Click - “Cast”, projectile attack
E - Interact with objects or other prompts
Q - “Special”, attack that damages enemies in a circle around you

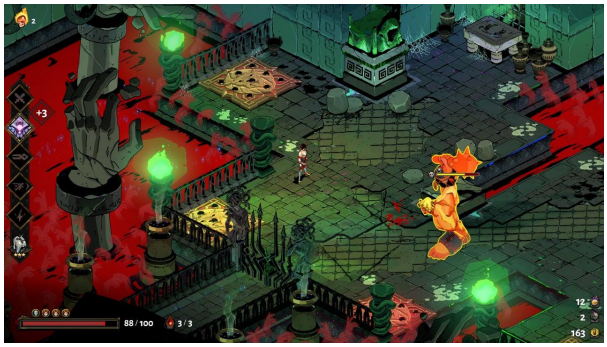
General Tips
Your overall goal is to proceed through the different rooms by defeating all enemies in each room.

There are trap panels on the floor which will damage you if you step on them after a delay.

When you’ve cleared all enemies from room, you proceed to the next room by interacting with the doorways with orbs.

a

Some enemies will have armour which makes them immune to stun while they attack. Once you damage them enough, the armour will go away and they will behave like normal enemies.



Hollow Knight

Arrow keys - Move your character
Z - Jump
X - Attack
A - Press and hold this once your meter glows white to heal

Unfilled meter with health at 2/4

Usable amount of meter

Note that unlike many other games, your jump stops rising *immediately* if you let go of the jump button, so only let go if you have safe ground directly below you

Bloons Tower Defense 6

Mouse - Move cursor, select menu items
Esc - Pause hotkey
Space - Fast-forward hotkey

Bloons Tower Defense 6 is a tower-defense game where you place down “towers” (in this case, monkeys) to prevent “invaders” (in this case, balloons) from progressing through a set path on the screen

Every time a balloon makes it to the end of the path, you lose one life (counted with the heart in the upper-left corner). If this counter reaches zero, the game ends.

Towers cost money, which is earned by popping balloons. Money can also be used to upgrade existing towers (you can do this by clicking on the tower you want to upgrade), though placing new towers (which is done via the menu on the right) may be necessary since each one can only attack with a certain frequency in a certain range.

Consent Form

Informed Consent Agreement for Participation in a Research Study

Investigators: Carter Nakagawa, Michael Oliveira, Thomas Flanagan
Contact Information:
Professor Mark Claypool - claypool@wpi.edu
Carter Nakagawa - clnakagawa@wpi.edu
Michael Oliveira - mjoliveira@wpi.edu
Thomas Flanagan - tmflanagan@wpi.edu
Title of Research Study: IQP - Latency and Jitter for Google Stadia Games
Sponsor: Professor Mark Claypool

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 the study:
The goal of this study is to measure the effects of latency and jitter on player experience in cloud gaming systems. We aim to investigate how different degrees of jitter frequency and magnitude impact the overall quality of experience for players.

Procedures to be followed:
During this study you will be expected to play 4 different games on a desktop computer for a set amount of time (~12-15 minutes per game). For each game, you will have a 1-2 minute period to practise the basic controls of the game before we start collecting data. The gameplay will be divided into 10 “rounds”. Between rounds, we will pause the game and ask you questions about your experience. The experiment will take a total of about 1 hour to complete.

Risks to study participants:
Video games have a risk of causing seizures due to potential flashing lights and other seizure inducing visuals. This risk is only applicable to those with a history of seizures. If you have a history of seizures, you are advised to not participate in this study.

The 4 games that you will be asked to play are:
Bloons TD 6 - ESRB rated E, no associated warnings.
Hollow Knight - ESRB rated E 10+, contains fantasy violence, mild blood, and insect-like creatures.
Hades - ESRB rated T, contains alcohol Reference, blood, mild language, suggestive Themes, and violence.
Counter Strike: Global Offensive - ESRB rated M, contains blood and intense violence.

If you are uncomfortable with the content of any of these games, you are advised to not participate in this study.

Benefits to research participants and others:
None.

Record keeping and confidentiality:
For this study, we will record basic demographic information (age/gender) as well as your answers to survey questions. Any interaction with the desktop computer used for testing may be recorded. Records of your participation in this study will be held confidential so far as permitted by law. However, the study investigators, the sponsor or its 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 or treatment in the event of injury:
There is minimal risk associated with this study. Please note that you do not give up any of your legal rights by signing this statement.

Payment:
All participants will be compensated with one $15 Amazon gift card. All participants will be given 2 IMGD playtesting credits.

For more information about this research or about the rights of research participants, or in case of research-related injury, contact:
Researchers - See contact information at the beginning of this document.
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

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.

_____ _____

Participant Signature Date

_____

Participant Name (Please print)

_____ _____

Signature of Person who explained this study Date