Analyzing the Impact of Frametime Spikes on Navigation-Based Tasks in 2D Platformers

A Major Qualifying Project Report

Submitted to the Faculty of

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Bachelor of Computer Science

by:

Nathan Anderson

Jonathan Asher

Connor Chartier

Date: 03/7/2025

Report Submitted To:

Professor Mark Claypool

Worcester Polytechnic Institute

This report represents the work of one or more WPI undergraduate students submitted to the faculty as evidence of completion of a degree requirement. WPI routinely publishes these reports on the web without editorial or peer review

Abstract

The impact of frame rate on player performance and quality of experience (QoE) is well-studied in games, particularly in first-person shooter (FPS) games. In general, higher frame rates lead to better player performance and higher player QoE. However, the impact of frametime spikes on player performance and QoE has not been well-researched in navigation-based tasks. Our team modified SuperTux Classic, an open-source 2D platformer, to implement frametime spikes triggered at specific times during gameplay. Then, user study participants played the modified game while completing individual navigation-based tasks. Each round had a set frametime spike size for 8 different tasks, and the 31 users played through each task with each frametime spike size. We found that as frametime spike size increases, QoE scores decrease over all tasks. However, the relationship between performance and frametime spike size was dependent on the individual task. Frametime spike size did not significantly affect performance on combat and collectible tasks, while there was a clear negative correlation for jumping-related tasks.

Acknowledgements

Our team would like to acknowledge our advisor, Professor Mark Claypool, for his unwavering support, insightful guidance, and invaluable expertise. We are also immensely grateful to our graduate student mentor, Samin Tokey, for generously sharing his plentiful research experience, which greatly enhanced our project. Finally, we sincerely thank the Esports Research Team at NVIDIA for sharing their rich industry expertise and thoughtful feedback, which provided us critical perspectives and refinements.

Table of Contents

Abstract…… i
Acknowledgements……..ii
Table of Contents……iii
List of Figures……vi
List of Tables…… viii
1. Introduction……..1
1.1 Report Structure……3
2. Background……..5
2.1 SuperTux Classic……... 5
2.1.1 Coyote Time……6
2.2 Godot……..6
2.3 Frame Stutters…… 7
3. Related Works……..8
3.1 The Effect of Frame Rate on Player Performance……8
3.2 The Effect of Frame Rate on Player Quality of Experience…… 11
3.3 Generic Rubrics for Measuring Player Skill……13
3.4 Navigation-Based Tasks……...14
4. Methodologies……... 16
4.1 Choosing a Genre……16
4.2 Choosing a Game……16
4.3 Task Difficulty Rubric…… 16
4.3.1 Actions and Tasks…… 17
4.3.2 Precision/Deadline/Impact…… 18
4.4 Frametime Spike Design……..19
4.4.1 Frametime Spike Implementation……19
4.4.2 Expected vs. Measured Lag…….. 20
4.4.3 Delay Magnitudes……21

4.5 Experiment Design……..……..22
4.6 Level Design……..……... 24
4.6.1 Resetting Rounds……..…… 24
4.6.2 Creating the Round Levels……..……...24
4.6.2.1 Collect Moving Item……..……25
4.6.2.2 Collect Bouncing Item……..……... 26
4.6.2.3 Squish Bouncing Enemy……..……...27
4.6.2.4 Large Gap, Large Platform……..……... 28
4.6.2.5 Small Gap, Medium Platform……..……... 29
4.6.2.6 Medium Gap, Small Platform……..……... 30
4.6.2.7 7-Jump……..…….. 32
4.6.2.8 Z-Jump……..……..……. 34
4.6.3 Creating the Practice Levels……..…… 36
4.7 The Playtesting Harness……..……..36
4.7.1 User ID……..……..……36
4.7.2 Round Order……..……..……37
4.7.3 Player Spawn Positions……..……37
4.7.4 Background Distractions……..……...37
4.7.5 Popups……..……..……38
4.7.5.1 Pre-Round Popup……..……..……38
4.7.5.2 Post-Round Popups……..……... 40
4.7.6 Player Score……..……..…… 41
4.7.8 Round Counter……..……41
4.8 Logging System……..……..…… 41
4.8.1 Log Types……..……..……42
5. Results……..……..……..44
5.1 Outlier Analysis……..……..…… 44
5.2 Data Analysis……... 45
5.2.1 Performance…….. 45
5.2.2 Quality of Experience……... 52

5.3 Precision/Deadline/Impact……...60
5.3.1 Effectiveness at Predicting Player Performance……... 60
5.3.2 Effectiveness at Predicting QoE Scores……64
5.3.3 Effectiveness at Predicting Frametime Spike Sensitivity……. 66
5.3.4 Average Precision/Deadline/Impact Score as a Predictor of Success Rate and QoE... 70
5.4 Summary……... 72
6. Conclusion……. 72
7. Future Works……..74
7.1 Limitations……... 74
7.2 Challenges……75
7.3 Additional Research……76
7.3.1 Short-Term…… 76
7.3.2 Medium-Term……... 77
7.3.3 Long-Term…… 78
References……..79
Appendices……80

List of Figures

Figure 1: Tux jumping…… 6
Figure 2: Task: Tux Jumping Over a Gap……..17
Figure 3: Actions Required To Complete the Task (Figure 2).....18
Figure 4: Frametime Spike Locations in the “Small Gap, Medium Landing Platform” Round…20
Figure 5: The “Collect Moving Item” Level……..26
Figure 6: The “Collect Bouncing Item” Level……...27
Figure 7: The “Squish Bouncing Enemy” Level……... 28
Figure 8: The “Large Gap, Large Platform” Level…… 29
Figure 9: The “Small Gap, Medium Platform” Level.……...30
Figure 10: The “Medium Gap, Small Platform” Level.……..31
Figure 11: The “7-Jump” Level.…… 33
Figure 12: The “Z-Jump” Level.……35
Figure 13: A Fish in the background of the level.…… 37
Figure 14: Figure 14: Pre-round popup.…… 38
Figure 15: The practice level pre-round popup.……..39
Figure 16: Post-round QoE score popup.……...40
Figure 17: Post-round QoE acceptability popup.……...40
Figure 18: Player info GUI.……... 41
Figure 19: Practical Success Rate vs. Frametime Spike Size (ms).……...46
Figure 20: Practical Success Rate vs. Task.……... 47
Figure 21: Practical Success Rate vs. Spike Time Per-Task Grouping.……..48
Figure 22: Comparison of Trendlines in Figure 21……48
Figure 23: Actual Success Rate vs. Spike Time Per-Task Grouping.…… 49
Figure 24: Actual and Practical Success Rate vs. Spike Time Per-Task Grouping.…….. 50
Figure 25: Actual and Practical Success Rate vs. Spike Time Per-Task.…….. 50
Figure 26: Practical Success Rate vs. Frametime Spike Size (ms) Per-Task.……51
Figure 27: Comparison of Practical Success Rate vs. Spike Time Per-Task (ms).……52
Figure 28: Mean QoE Score vs. Frametime Spike Size (ms).……... 53

Figure 29: QoE Score vs. Frametime Spike Size (ms) Per-Task Grouping.……..54
Figure 30: Distribution of QoE Scores Per-Spike Size.……55
Figure 31: Violin Plot QoE Score vs. Frametime Spike Size (ms).……...55
Figure 32: Average QoE Score vs Spike Time.…… 56
Figure 33: Average QoE vs Spike Time Per-Task.…… 57
Figure 34: Distribution of QoE Scores.…… 57
Figure 35: Distribution of Mean QoE Scores.……... 57
Figure 36: Distribution of QoE Scores Per-User.…….. 58
Figure 37: Acceptability Rate vs. QoE Score.……... 59
Figure 38: Average Practical and Actual Success Rate vs. Precision Score.……60
Figure 39: Average Practical and Actual Success Rate vs. Deadline Score.…… 61
Figure 40: Average Practical and Actual Success Rate vs. Impact Score.…… 61
Figure 41: Average Practical Success Rate vs. Precision/Deadline/Impact Score.……... 62
Figure 42: Average Actual Success Rate vs. Precision/Deadline/Impact Score.……...63
Figure 43: Average QoE Score vs. Precision Score.……..64
Figure 44: Average QoE Score vs. Deadline Score.…….. 64
Figure 45: Average QoE Score vs. Impact Score.…… 65
Figure 46: QoE Score vs. Precision/Deadline/Impact Score.…… 66
Figure 47: Actual Success Rate vs Precision Per-Frametime Spike Duration.……..67
Figure 48: Actual Success Rate vs Deadline Per-Frametime Spike Duration.…….. 67
Figure 49: Actual Success Rate vs Impact Per-Frametime Spike Duration.……. 68
Figure 50: Practical Success Rate vs Precision Per-Frametime Spike Duration.…….. 68
Figure 51: Actual Success Rate vs Deadline Per-Frametime Spike Duration.…….. 69
Figure 52: Actual Success Rate vs Impact Per-Frametime Spike Duration.……. 69
Figure 53: Actual Success Rate vs. Averaged Precision/Deadline/Impact Score.……70
Figure 54: Practical Success Rate vs. Averaged Precision/Deadline/Impact Score.……. 71
Figure 55: Average QoE Score vs. Averaged Precision/Deadline/Impact Score.……..71

List of Tables

Table 1: Expected vs. Measured Lag……21
Table 2: The Minimum and Maximum Frametime Spike Delay for all Rounds……...22
Table 3: The Different Objectives for Each Level……39

1. Introduction

Computer components have seen a lot of improvement even since just 10 years ago, and with that comes the opportunity for game developers to make their games look even more stunning. From titles such as the original Doom, Minecraft, and a more recent era of Cyberpunk 2077, game developers have been improving the graphical aspect of their games as hardware has improved – using ray tracing, higher render distance, and more detailed textures.

However, these graphics improvements can come at a cost. Cyberpunk 2077 is notorious for having frame rate and frame drop issues if it is not being played on a machine with the latest hardware (CD Projekt Red, 2020). Low frame rate is becoming somewhat of a normal occurrence in more modern titles, and frame drops are an increasingly common issue as load screens are removed from final products. This causes more of the game world to be loaded and rendered at any given time. This can have negative implications on the performance of a given game, however, it provides a more immersive experience for the player.

Cloud-based games add latency due to the client having to receive in-game information from a server over a network, and network latency can behave similarly to frame stutters. When there is high latency sending data across the network, frames are delivered late, causing a visual stutter. Minimizing network latency is important for cloud-based platforms to potentially improve a player's performance and quality of experience when using the platform. Understanding the impact of frame stutter on player performance and quality of experience in games may help design better cloud-based platforms.

The impact of frame rate on player performance and quality of experience (QoE) is well-studied in games, particularly in first-person shooter (FPS) games. In general, higher frame

rates lead to better player performance and higher player QoE (Claypool K. & Claypool M., 2007).

While various impacts on player performance are studied well within the first-person shooter genre, they are much less studied in other categories. One specific task in video games -navigation- has not been researched as to what can affect player performance and QoE. Despite being an area that has little research, many game genres feature navigation as a core part of their gameplay, which gives potential to navigation research to have a high impact. FPS games have components of navigation in their gameplay, but the main focus of studies in FPS games has been concentrated on aiming and shooting mechanics.

Unlike FPS games, 2D platformers are a genre of games that primarily revolve around navigation. The generic gameplay of a 2D platformer consists of trying to get from point A to point B while avoiding obstacles in the way, which is a navigation-dominant task.

While the impact of frame rate on player performance and QoE is well understood, the impact of frame stutters is not. Studies on frame stutters have been limited to game-specific actions and general gameplay QoE, and there has been little research into the impact of frame stutters on player performance and QoE in navigation-based tasks in games. The purpose of this research is to provide insight into the impact of frame stutters on navigation-based game tasks, which are common in many game genres.

Our team created a task difficulty rubric to rank the difficulties of different tasks and decided to implement frametime spikes using the “Random Delay” method, where exiting a frametime spike trigger causes the frametime spike after a random delay (usually between 10 and 69 ms). We also decided on four different frametime spike durations of 0, 75, 150, and 225 ms.

Our study consists of the participant playing two 90-second practice rounds, where the first has a 0 ms (best-case) frametime spike duration, and the second has a 225 ms (worst-case) frametime spike size. The participant then plays 32 playtesting rounds, where each round has the participant complete the same task as many times as they can within the round time (usually 30 seconds). Each round has a set frametime spike size, and there are 8 different tasks in total. The player plays through each task with each frametime spike size.

Through analyzing our data we can deduce that as frametime spike size increases, QoE scores decrease over all task groupings. However, the data show that the relationship between participant performance and frametime spike size is dependent on the individual task grouping. For the enemy and collectible groupings, performance did not have a significant change when frametime spike time increased, whereas, with jumping-related tasks there was a clear negative correlation. When revisiting our precision/deadline/impact rubric, it was apparent that it was a significantly better predictor of a task's sensitivity to frametime spike size as it relates to performance rather than for QoE.

The rest of the report is as follows: Chapter 2 describes relevant background content, such as SuperTux Classic and coyote time; Chapter 3 describes the methodologies and findings of other studies on the impact of frame stutters on player performance and quality of experience; Chapter 4 describes our methodologies in choosing a genre, game, creating a rubric to classify task difficulty, our design and implementation of frametime spikes, our design of the experiment, level design, and the design of the playtesting harness and logger; Chapter 5 covers data analysis, including outlier detection, the effect of frame stutters on player performance and quality of experience, and the effectiveness of our rubric at predicting task difficulty and sensitivity to

frame spikes; Chapter 6 is our conclusion; and Chapter 7 covers our limitations and challenges in running the study as well as suggested future research.

2. Background

This section covers information that is required to understand parts of the report, as well as other information such as movement parameters in SuperTux Classic.

2.1 SuperTux Classic

SuperTux Classic (Alzter, 2023) is an open-source Mario clone with a level editor. It is a 2D platformer, which is a game genre where the player controls a character to reach the end of a level by navigating through jumping puzzles. The character the player controls in SuperTux Classic is called Tux.

Each tile in SuperTux Classic is 32 pixels wide and 32 pixels tall. The following parameters use units of both pixels and tiles to represent their in-game values. Tux has a jump height of 5 tiles when standing still or walking and 6 tiles when running. Tux is determined to be running if his horizontal speed is greater than 10.35 tiles/sec. Tux's maximum walking speed is 2.3 * 4.5 (10.35) tiles/sec, with a walking acceleration of $\frac{0.03*60^2}{8}$ (13.5) pixels/sec^2. When Tux first starts walking, his speed is set to 0.5 pixels/sec. Tux's maximum running speed is 3.2 * 4.5 (14.4) tiles/sec, with a running acceleration of $\frac{0.04*60^2}{11}$ (~13.09) pixels/sec^2. When Tux first starts running, his speed is set to 4.5 pixels/sec. Tux's walk friction – the rate at which he slows down while the player is not moving when he is at walking speed – is $\frac{0.03*60^2}{8}$ (13.5) pixels/sec, and his run friction is $\frac{0.03*60^2}{4}$ (27) pixels/sec.

2.1.1 Coyote Time

Coyote time is a leniency mechanic in games that lets players jump for a short time after they walk or run off of a ledge. SuperTux Classic has 100 ms of coyote time, which at 60 FPS allows Tux to jump for up to 5 frames after walking or running off a ledge. At maximum running speed, this amounts to 1.2 blocks, which is shown in Figure 1.

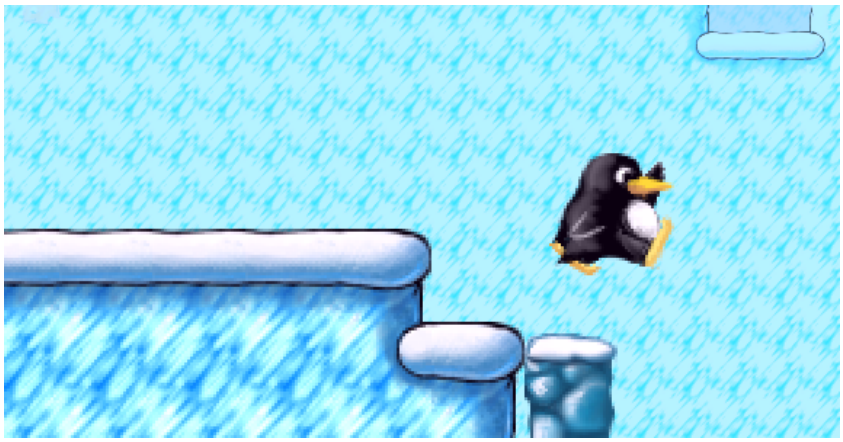


Figure 1: Tux Jumping. The last frame of coyote time after running off the left ledge at full speed. At this point, Tux has moved 1.2 blocks away from the ledge.

2.2 Godot

SuperTux Classic uses the Godot 3.5.2 (Godot, 2023) engine to run. Godot is also open source, however, this study did not need to manually edit core engine code. Godot games can be exported to most operating systems, including Windows, Linux, MacOS, and HTML for running in browsers. Games built in Godot use GScript as their scripting language, which is an interpreted language based on Python. Using this language, Godot abstracts and exposes many OS level system calls that were used in this study, such as fetching the machine clock time.

Godot is also multithreaded, with threads for rendering, physics, and audio (*Godot's Architecture Diagram*, 2024).

2.3 Frame Stutters

Frame stutters are a common occurrence in many video games. For a game to run at 60 frames per second consistently, it must finish all calculations for a given frame in less than 16 and 2/3rds milliseconds. Should a computer be unable to finish all calculations in that amount of time, whether due to a sudden increase in the amount of work needing to be done on a frame, delay in processes due to the system, a frame stutter, or a sudden drop in frames per second, will occur. Apart from just a visual stutter, inputs can be missed during the drop in frames as a game can only update once per frame. In this study, frametime spikes interact with inputs through a catchup method, courtesy of Godot being multithreaded. All inputs during a spike are stored, and then on the next actionable frame they are fast-forwarded through as though they all happened in order on that frame.

If the player is in the middle of a jump and releases then re-presses jump during a frametime spike, the jump will act as if it was released and the player will not gain any more height from the jump. If the player is grounded and presses and releases the jump button while the frametime spike is active, the jump will not register and the player will remain grounded.

3. Related Works

Our study investigates the impact of frame stutters on player performance and quality of experience. There have been previous studies conducted that have also explored this gap in research. These related works serve as the foundation of our understanding of this state of the art and how to conduct a study with a similar problem statement.

3.1 The Effect of Frame Rate on Player Performance

Claypool and Claypool evaluated the impact of frame rate on player performance in First Person Shooter games (Claypool, K., & Claypool, M., 2007). This study is also aimed at evaluating how frame rate impacts perceived quality with active gameplay as opposed to watching a video stream. This study was conducted with nearly 100 participants playing *Quake 3 Arena*. *Quake 3 Arena* also allowed for command line control over display parameters, which is helpful when modifying game frame rates. Three custom maps were made to test player movement and shooting actions. The pre-gameplay survey consisted of demographic questions and previous gaming experience. Five different frame rate values were tested: 3, 7, 15, 30, and 60 FPS. Three different resolutions were also tested: 640 × 480, 512 × 384, and 320 × 240 as these were common resolutions between PCs and for handheld consoles. There was also a single configuration that featured the highest frame rate (80) and resolution (1024 x 768), which all users played as their first game. The game was managed using a client program to handle the sequence of games and to record the qualitative user input after each round. There was also another program that was used to log the in-game events and statistics during gameplay. One complete run through the experiment took about 10 minutes for each participant. Users were

offered incentives for participating in the study, such as a gift card raffle, course credit, and refreshments. There were 36 participants for the movement study and 64 participants for the shooting experiment. One measurement of performance in the movement experiment was the speed at which a participant reached the end of a map. Another was evaluating the amount of health lost due to veering off the defined game path into lava. The study showed that the time it took to complete the level decreased as the frame rate increased. Similarly, as the frame rate increased the amount of health lost during the round decreased. The most significant decline for all measurements of performance was the drop from 7 to 3 FPS. As for the shooting experiment, performance was measured in the number of times the user killed a bot. The experiment found that scores increased as the frame rate increased, with the largest jump in score being the jump from 7 to 15 FPS. It was found that actions that required precision and quick reaction, like aiming and shooting, were heavily impacted as the frame rate declined.

Our study also includes a comprehensive testing harness that includes level configuration loading through a file, logging system that captures statistics and in-game events. This study inspired our inclusion of a quality of experience survey to evaluate how participants perceived the hindrance at different frame latency values. Finally, our study adopted a pre-playtesting demographics survey that, similar to this study, evaluates participants' previous experiences with playing video games.

Claypool and Claypool conducted a study evaluating the performance and quality of experience for both navigation and aiming tasks in a custom game made in Game Maker v6 (Claypool, M., & Claypool, K., 2009). This experiment also featured an evaluation of how frame rate and resolution impact player actions across first-person linear, third-person linear, and third-person isometric in-game perspectives. This study also featured a test harness that handled

the demographics survey, sequence of rounds, qualitative feedback, and logging system to capture gameplay metrics. There were 27 participants in this study, who were tasked with navigation and shooting tasks in different in-game perspectives with a variety of frame rates. In the shooting experiment, the goal was to shoot enemy sprites and performance was measured by a participant's accuracy. For navigation, the goal was to move around the level to correctly identify a sword object from five other dummy objects. Performance in the navigation task was evaluated by the number of correctly identified swords in the given time. There were a total of 36 game sessions, with two of the sessions being practice rounds which had the user familiarizing themself with the navigation and the shooting tasks. Each round lasted 15 seconds and at the end of each round users were prompted with a post-round quality of experience survey regarding visual quality and playability. The navigation experiment results suggest that there is an increase in score as the frame rate increases for both third-person linear and first-person linear perspective, but no increase or decrease could be determined with the third-person isometric perspective. As for the shooting study, there was a strong increase in score as the frame rate increased, with significant increases in scores from 7fps to 15fps and 15fps to 30fps, respectively, for all perspective types. The study found that frame rates significantly impact in-game performance as opposed to the resolution of the game, which did not. It was also found that the impact of the frame rate on performance was not the same for aiming and navigation. This study informed our demographics survey to include questions about previous gaming experience as well as inspire our post-round QoE survey to evaluate how the player perceives the quality of the round. The testing harness in our study also includes many of the same features as the harness in this study including the handling of the post-round QoE survey.

3.2 The Effect of Frame Rate on Player Quality of Experience

Liu et al. conducted a 33-person user study of three PC games to identify the impact of variations in frame rates on a player's quality of experience (QoE) (Liu et al., 2023). This can help players make more informed decisions on computer system hardware and upgrades, and display settings. Also, understanding the relation between variations in frame rate and QoE could help game developers better optimize display functionalities in games to provide a better experience. Finally, evaluating this relationship can better inform the cloud-based game space which relies on game streaming which currently has innate variations in frame rate depending on the quality of the network and delivery rates. The experiment for this study features three games: Rocket League, Strange Brigade, and Valorant. Rocket League and Strange Brigade share a third-person gameplay perspective whereas Valorant is a first-person perspective game. In each game, the user is tasked with playing in a 1v1 against a computer opponent (bots). The gameplay was conducted in a single area with waves of bots for the user to shoot and kill. All three games were tested using two target frame rates: 60Hz and 120Hz. Frame variations were induced using counters of the Fibonacci sequence. A pilot study was conducted with 3 volunteers which helped fine-tune the amount of frame rate variation for the final study, the number and length or rounds, and gameplay instructions. Participants played one practice round with no counters, There were four levels of variation: the first featured no frame rate variation, the second had "noticeable" variation, the third was "mid" with significant variation that was still playable, and fourth was "severe" with nearly unplayable variations in frames. Each round lasted one minute and was preceded by a post-round QoE survey that consisted of rating the gameplay experience on a scale of 1.0 - 5.0 and a follow-up yes/no question regarding the acceptability of the gameplay experience. The study found that the average frame rate was not a reliable indicator of QoE for a

player. The variation in frame rate was also crucial to consider when predicting QoE as even when the average frame rate is high, as the time between frames increases the QoE tends to be negatively impacted. This study helped structure our study's post-round QoE survey with questions that succinctly prompt the user to describe how they perceived the round on a 1.0-5.0 scale and also whether or not that particular amount of variation is acceptable.

Xu and Claypool also conducted four individual experiments, two of which attempted to uncover the severity of frame jitter on player QoE outside the context of cloud-based network latency which is similar to our study (Xu & Claypool., 2024). Similar to the previous study, understanding the relationship between QoE and frame time variation can help players make more informed decisions on computer system hardware and upgrades, and display settings. This could also lead game developers to improve display performance in games to provide a better experience for players. The third of four studies utilized three games: Rocket League, Strange Brigade, and Valorant respectively. In these games, players were assigned tasks/missions to complete in each game as frame jitters were introduced during gameplay. This was accomplished using a Fibonacci sequence counter to increase CPU load causing reduced frame output in-game. These frame time variations came in four separate forms: "perfect" which meant there was no frame time variation, "low", "medium", and "high". Each round lasted a minute and participants spent about 60 minutes in the study. Strange Brigade and Valorant had shooting tasks whereas in Rocket League, the goal was to score goals against an AI opponent. There were two target frame times for this study: 60Hz and 120Hz. The fourth study utilized an open-source racing game named SuperTuxKart and players were tasked with completing a race against AI opponents. Each round lasted one minute on a single, fixed map. In this experiment, frame jitter was introduced using sleep time within the game loop. The game loop sleep lasted for either 0ms,

30ms, 115ms, or 200ms, depending on the round setting. The engine ran at a fixed rate of 60Hz. These studies found that the standard deviation of frame times was a good predictor of QoE for the player, it was also found that average frame rate was not an accurate predictor of QoE. This study inspired our team to look into the SuperTux series, a collection of open source games, for a prospective game within our study. Our study ended up deciding on SuperTux Classic for our experiment.

3.3 Generic Rubrics for Measuring Player Skill

Claypool and Claypool analyze different player actions, specifically actions in first-person shooters, and theorize that actions can be mapped along two axes: precision and deadline (Claypool, M., & Claypool, K., 2006). In this scheme, precision is considered the input accuracy necessary for the player to complete a given action whereas deadline is the time required to complete a given action. This mapping allows for the amount of player performance that will be impacted during any given action by network latency to be predicted, so long as the action can be mapped to these two axes. The ability to create accurate predictions is helpful to game developers to be able to accurately determine where compensation techniques will be maximally beneficial, network designers to be able to create better infrastructure for online games, and game players to make better choices about their internet connection. This model was used as the basis for the precision-deadline-impact model that is used in this study and allows for the compartmentalization of actions within SuperTux Classic.

3.4 Navigation-Based Tasks

Liu and Claypool explore the impact of local latency vs. network on player performance and quality of experience in a two-player hide-and-seek game, a game that primarily focuses on movement (navigation) in a 30-person study (Liu & Claypool, 2022). This study identified a gap in research in navigation-based tasks and their sensitivity to local or network latency. The goal of the game was for the player to remain undetected as the hider and to spot the other player as the seeker. The roles of hider and seeker would change every 2-6 seconds to simulate the phenomenon of being the “hunter” and “hunted” within a first-person shooter game. Each round lasted 40 seconds. Performance was determined by the number of points a player had which was calculated every frame. A player could earn a point in two ways: Firstly, each frame the seeker had the opponent in their view, secondly, each frame the hider was not in the view of the seeker. The game engine ran at 50fps and each frame was logged to record each point awarded. Each participant played against the same human opponent to ensure consistency and realism when compared to an AI-controlled opponent. Furthermore, network and local latencies were only applied to the participant’s avatar. Players had one practice round with no latency before playing 27 rounds at different local latencies (25ms, 100ms, 175ms) and network latencies (0ms, 100ms, 200ms). Each round the user was prompted with a one-question QoE survey to complete where participants rated the quality of the round on a 5-point Likert scale. After all rounds were completed, users completed a post-round demographics survey regarding previous gaming experiences. The results found that there was no significant difference in performance between the two forms of latency. However, performance and quality of experience decreased as the latency increased for both local and network latency. Additionally, QoE scores decreased as latency increased. It was identified in this paper that there is more research to be done to

understand the impact of latency on individual actions within games. This study was an inspiration to explore the impact of frame latency on player performance and experience in navigation-based tasks. This study further justifies the use of a post-round QoE survey with potential conclusions to be drawn from players' qualitative experiences playing with latency.

4. Methodologies

This chapter covers the methods we used and the decisions we made when designing and running the study.

4.1 Choosing a Genre

We chose the 2D platformer genre for a few reasons. First, 2D platformers are a navigation-focused genre, which is important for our study. 2D platformers are also a simple and well-known genre, which makes them easier for players to learn. There are many open-source 2D platformers, which lets us choose the game that best fits our requirements. Finally, 2D platformers are generally not resource-intensive, so there should be no unintended stutters from the system being overloaded.

4.2 Choosing a Game

We chose SuperTux Classic because it is open-source, which makes it easier to modify, and it is built in Godot, which has cross-platform support as well as a built-in performance profiler.

4.3 Task Difficulty Rubric

The task difficulty rubric provides a way of measuring the difficulty of tasks relative to each other.

4.3.1 Actions and Tasks

We defined an action as an input, combination of inputs, or release of inputs to modify the player's position or velocity. We defined a task as a sequence of no more than five actions with a clear objective. An example task is jumping over a gap, shown in Figure 2. The player controls Tux, the penguin avatar, and presses space to jump while moving from the right to the left.



Figure 2: Task: Tux Jumping Over a Gap.

This task consists of two actions: running towards the gap and jumping over the gap, which are shown individually in Figure 3.

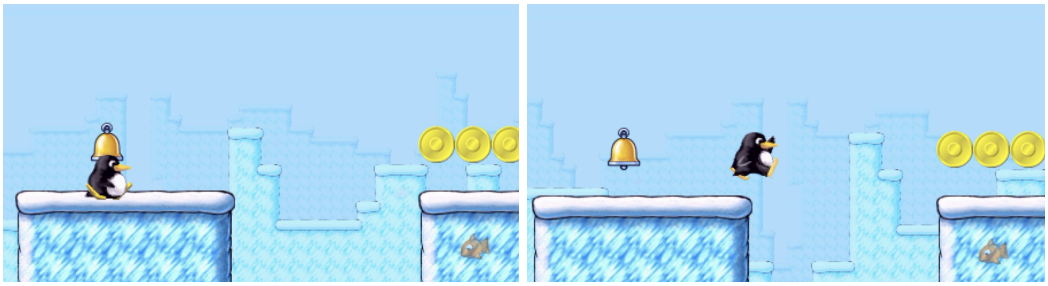


Figure 3: Actions Required To Complete the Task (Figure 2). In the image on the left, the player is running to the right. In the image on the right, the player has just pressed the jump button while holding right and is jumping over the gap.

4.3.2 Precision/Deadline/Impact

We identified three categories that affect the difficulty of a task:

Precision [1 (Low) - 5 (High)]: The minimum number of actions required to complete the task. A task with a precision of 1 requires 1 input to complete successfully, and a task with a precision of 5 requires 5 inputs to complete. If a task requires more than 5 inputs to complete, it is split into multiple tasks, each of which requires 5 or fewer inputs to complete.

Deadline [1 (Low) - 5 (High)]: The shortest window of time for an input across all actions in the task. A task with a deadline of 1 has no window of time, and a task with a deadline of 5 has a very tight window of time.

Impact [1 (Low) - 5 (High)]: The setback that is incurred upon failing the task. A task with an impact of 1 has no setback, and a task with an impact of 5 has a severe setback (e.g. death).

4.4 Frametime Spike Design

This section covers the design and implementation of frametime spikes in the study.

4.4.1 Frametime Spike Implementation

For this study, we chose to go with a ‘Random Delay’ spike design. This type of frametime spike works by defining an area of interconnected cells. Entering one of these areas initializes a “delay” timer, which begins to count down once the player leaves the field. This timer stops counting down if the player ever stops, and gets reset when the player re-enters a field to prevent players from potentially “dodging” spikes. Once the timer hits zero, a spike is thrown. This spike design was chosen for how customizable it is, allowing it to apply to a wide variety of tasks. Figure 4 shows an example of where the frametime spikes are placed in a level. The player spawns on the tile with the white flag. The goal of this task is to jump over the gap and touch the bell. The triggers are the red boxes with “...” in the middle.

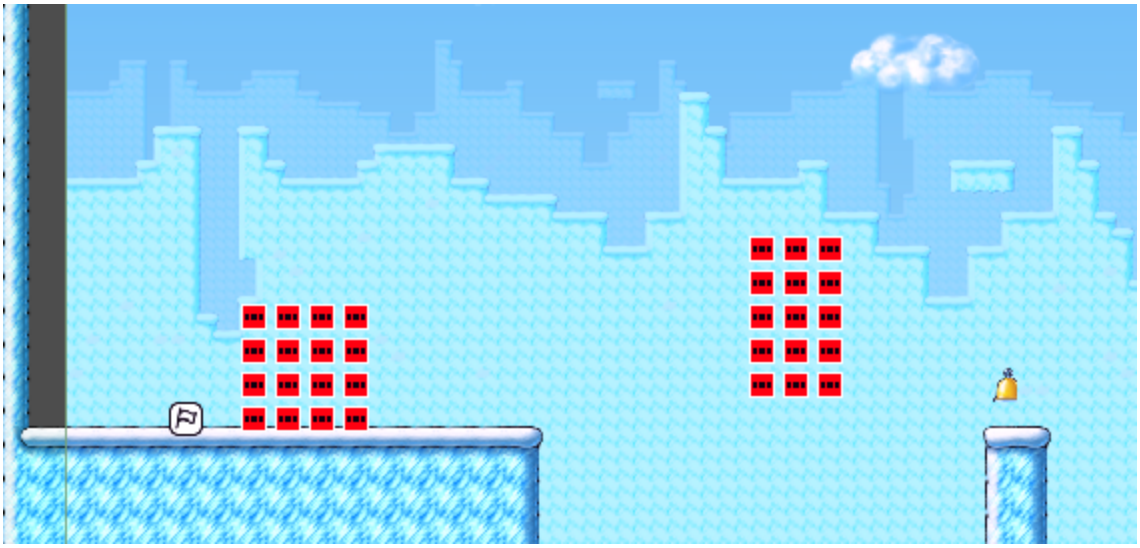


Figure 4: Frametime Spike Locations in the “Small Gap, Medium Landing Platform” Round. The first (left) set of triggers are placed to trigger a spike right before the player jumps, and the second (right) set of triggers are placed to trigger a spike right before the player starts to slow down.

4.4.2 Expected vs. Measured Lag

The expected lag is the duration that was delayed to cause a frametime spike. The measured lag is the duration that the frametime spike was measured to be. Table 1 shows a comparison of the differences in the expected and measured lag in our study. For each frametime spike duration, we measured five different spikes using a 1000 FPS camera and calculated the mean from the results (Appendix D). Every result in Table 1 was within one frame (16.667 ms) of the expected lag.

Expected Lag (ms)	Measured Lag (ms)	Difference (ms)	Standard Deviation (ms)
75	91.53	+16.53	2.18
150	164.4	+14.40	1.72
225	234.39	+9.39	1.90

Table 1: Expected vs. Measured Lag. The difference in the expected lag vs actual lag for the various stutters in the study.

4.4.3 Delay Magnitudes

The magnitude of the delay for each frametime spike is set on a per-level basis, and is set in an attempt to cause the frametime spike during the worst possible time in the task, with a little bit of randomness thrown in so that the spikes do not become predictable. This is done by calculating how long it will take the player to reach a “critical zone” – the area in which the player must jump to complete the task – and setting the maximum value of the delay to just under that duration. Table 2 shows the minimum and maximum delays for all the rounds.

Round Type	Delay Min (ms)	Delay Max (ms)
Practice	10	69
Small Gap, Medium Platform	10	207
Medium Gap, Small Platform	10	207
Large Gap, Large Platform	10	69
Collect Item: Moving	10	400
Collect Item: Bouncing	10	621
Squish Enemy	10	400
7-Jump	10	69
Z-Jump	10	69

Table 2: The Minimum and Maximum Frametime Spike Delay for all Rounds.

4.5 Experiment Design

While the impact of frame rate on player performance and QoE is well understood, the impact of frame stutters is not. Studies on frame stutters have been limited to game-specific actions and general gameplay QoE, and there has been little research into the impact of frame stutters on player performance and QoE on navigation-based tasks in games. To identify the impact of frame stutter on navigation-based tasks, our study was approved by the IRB and conducted in the following fashion (Appendix A, B, & E):

1. The participant is asked to be seated and make themselves comfortable in front of a computer with a keyboard and mouse.

2. The participant is prompted to review the consent form (Appendix F).
    a. If they accept and sign, we will continue the procedures.
    b. If they decide not to sign the consent form, they are free to leave.
3. The participant is asked to take a 2-3 minute demographics survey regarding previous gaming experiences.
4. The participant is asked to take a 10-round reflex test.
5. The participant plays (2) practice rounds, completing a two-question survey after each round.
    a. At the start of each practice round, a popup is shown explaining the controls and objective of the level. The popup disappears when any movement key is pressed. See Section 3.7.5.1 (Pre-Round Popup) for details on the popup.
    b. The first practice round is the smoothest possible experience (no frametime spikes) and the second practice round is the choppiest possible experience (225ms frametime spikes). This familiarizes the player with what a 1 and 5 are on the QoE scale.
    c. After each practice round, the player completes a two-question survey. See Section 3.7.5.2 (Post-Round Popups) for details on the contents of the survey.
6. The participant plays (32) playtesting rounds, completing a two-question survey after each round.
    a. At the start of each round, a popup is shown explaining the objective of the level. The popup disappears when any movement key is pressed. See Section 3.7.5.1 (Pre-Round Popup) for details on the popup.

   b. After each playtesting round, the player completes a two-question survey. See Section 3.7.5.2 (Post-Round Popups) for details on the contents of the survey.
   c. After the final playtesting round, a “Thank you” screen is shown and the playtesting is over.
7. The participant gathers their belongings and exits the playtesting room.

Each playtesting session lasts for approximately 25 minutes. The full script used for playtesting is attached in Appendix _ (TODO: Fill this in once we have appendices)

4.6 Level Design

This section covers the design of the rounds and the practice level.

4.6.1 Resetting Rounds

Some rounds require the player to jump over a gap. For these rounds, we created a reset bell, which resets the level when the player touches it. Some rounds have a small platform the player needs to land on. The bell only activates when the player is touching it with a velocity of under 100 units (just under walking speed) to ensure the player has successfully stopped on the platform.

4.6.2 Creating the Round Levels

There are 8 total playtest round levels designed for this study. Each level has a different layout and design to get data from multiple places in the precision/deadline/impact rubric. The

spike triggers are manually placed on each level. Each level has its custom setting for the length of the delay before a spike is thrown.

4.6.2.1 Collect Moving Item

The goal of this task is to collect (touch) an item moving away from the player at a constant speed before it falls into a gap. The player fails if the item falls 2 tiles below the top of the gap. If the player successfully collects the item, a bell sound is played, the score increases by 100 points, and the level is reset. The level timer for this task is 15 seconds. It is possible to complete this task 12 times in a single round.

The frametime spike triggers are positioned such that a frametime spike is triggered right before the player collects the item. The horizontal movement speed of the item is 3.2 tiles per second and the player's maximum movement speed is 14.4 tiles per second. The item spawns 8 tiles away from falling, and the player spawns 15 ± 1 tiles away from the item.

This task has a precision/deadline/impact score of 1/2/2. The frametime spike delay is 10-400 ms. Figure 5 shows the level layout for this task, which consists of the player spawn, the frametime spike triggers, and the collectible's spawn position.

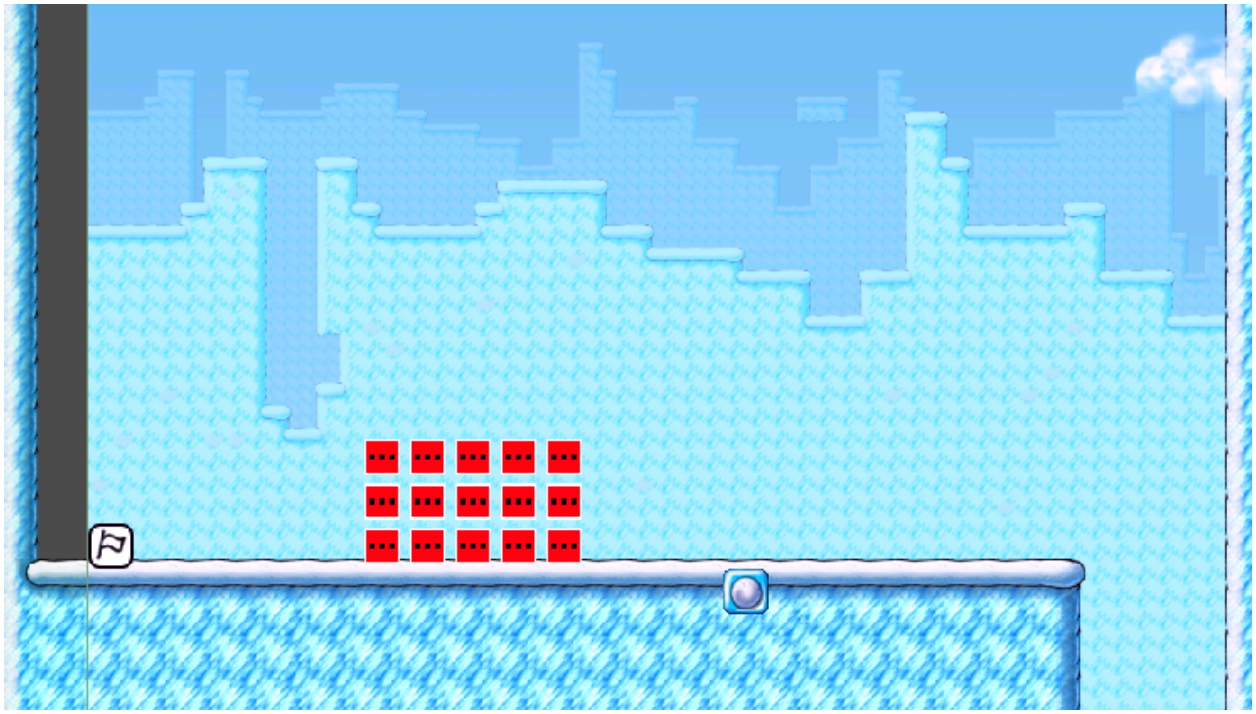


Figure 5: The “Collect Moving Item” Level. The item spawns from the block to the right of the frametime spike triggers and moves toward the gap on the right.

4.6.2.2 Collect Bouncing Item

The goal of this task is to collect (touch) an item that is bouncing away from the player before it falls into a gap. The player fails if the item falls 2 tiles below the top of the gap. If the player successfully collects the item, a bell sound is played, the score increases by 100 points, and the level is reset. The item moves at a consistent speed, bounces at a consistent height, and falls into the gap at the end of its second bounce. The level timer for this task is 15 seconds. It is possible to complete this task 7 times in a single round.

The frametime spike triggers are positioned such that a frametime spike is triggered right before the player collects the item. The horizontal movement speed of the item is 3.2 tiles per second, its vertical movement speed is 6 tiles per second right after a jump (the game’s gravity

decelerates it), and the player's maximum movement speed is 14.4 tiles per second. The item spawns 8 tiles away from falling, and the player spawns 15 ± 1 tiles away from the item.

This task has a precision/deadline/impact score of 2/3/2. The frametime spike delay is 10-621 ms. Figure 6 shows the level layout for this task, which consists of the player spawn, the frametime spike triggers, and the collectible's spawn position.

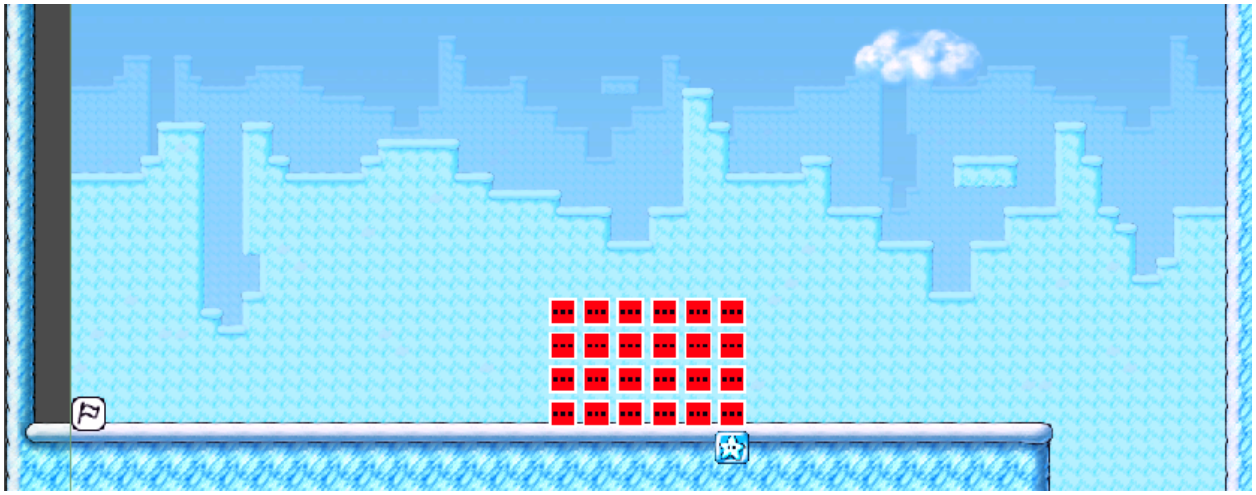


Figure 6: The "Collect Bouncing Item" Level. The item spawns from the block in the ground and bounces toward the gap on the right.

4.6.2.3 Squish Bouncing Enemy

The goal of this task is to jump on top of (squish) an enemy that is bouncing in a pit. The enemy moves at a consistent speed and bounces at a consistent height. If the player dies at the enemy, a death animation is played, which lasts for approximately 2 seconds. The level timer counts down during the death animation. The level timer for this task is 15 seconds. It is possible to complete this task 12 times in a single round.

The frametime spike triggers are positioned such that a frametime spike occurs right before the player lands on the enemy.

This task has a precision/deadline/impact score of 3/3/3. The frametime spike delay is 10-400 ms. Figure 7 shows the level layout for this task, which consists of the player spawn, the frametime spike triggers, and the enemy spawn position.

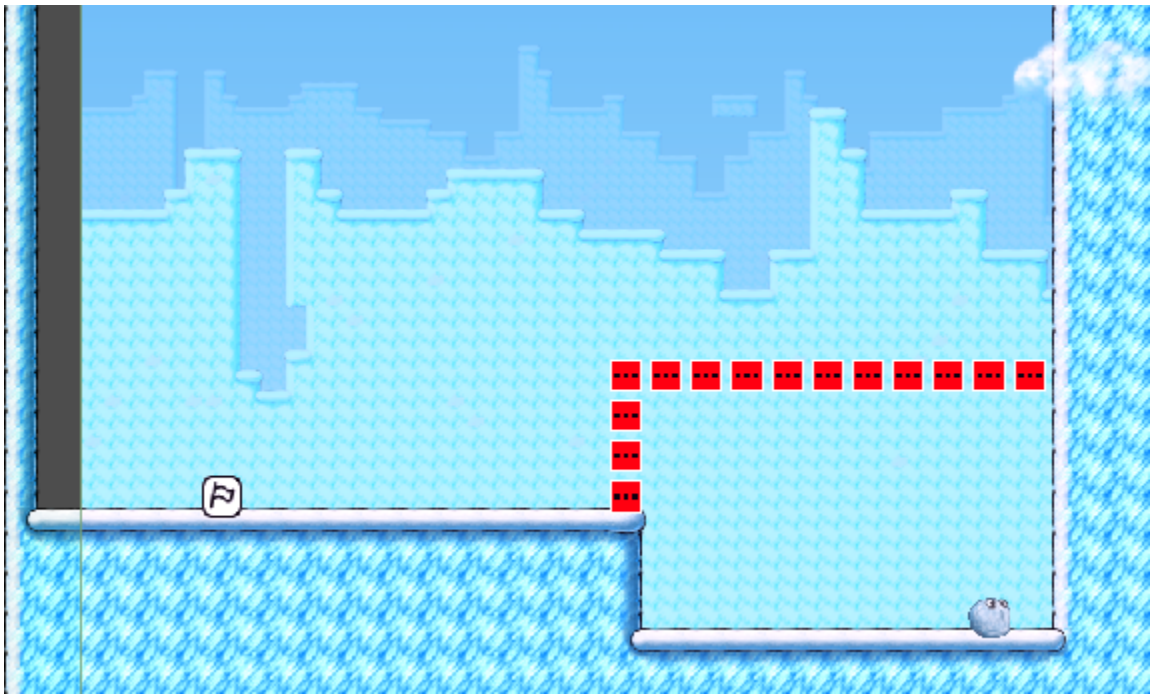


Figure 7: The “Squish Bouncing Enemy” Level.

4.6.2.4 Large Gap, Large Platform

The goal of this task is to jump over a large (17-block) gap and touch the bell on the other side. The “Large Platform” part of the task refers to the size of the platform the player lands on. If the player falls into the gap, a death animation is played, which lasts for approximately 1 second. Additionally, it takes approximately 1 second to fall to the bottom of the screen and play the death animation. The level timer counts down during the death animation. The level timer for this task is 30 seconds. It is possible to complete this task 13 times in a single round.

The frametime spike triggers are positioned such that a frametime spike occurs right before the player jumps. If the player jumps before or during a frametime spike, their jump will not go far enough to clear the gap.

This task has a precision/deadline/impact score of 2/5/5. The frametime spike delay is 10-69 ms. Figure 8 shows the level layout for this task, which consists of the player spawn position, the frametime spike triggers, and the bell.

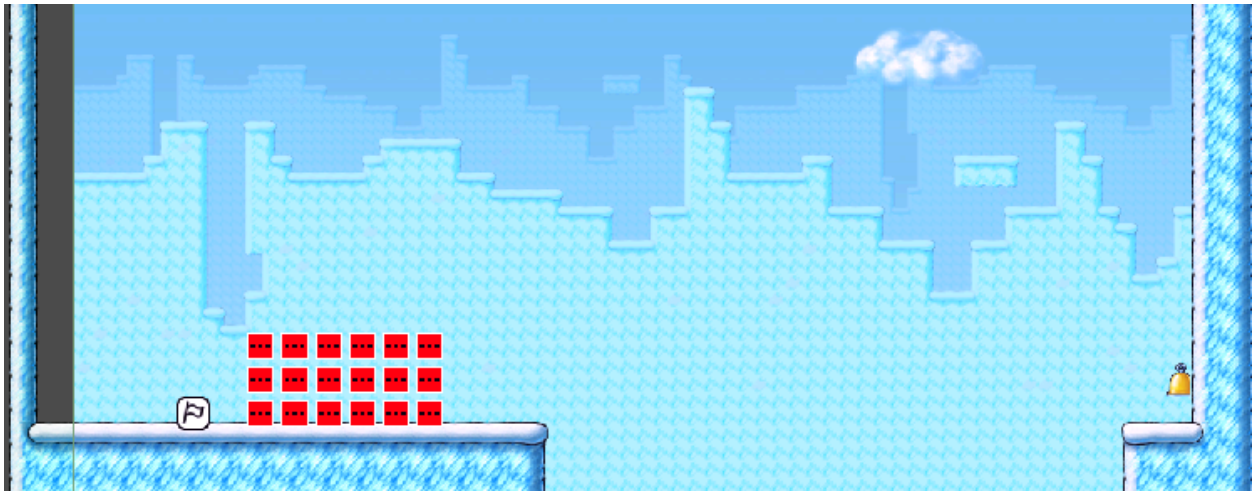


Figure 8: The “Large Gap, Large Platform” Level.

4.6.2.5 Small Gap, Medium Platform

The goal of this task is to jump over a small (12-block) gap and touch the bell on the other side. The player must land on a medium (2-block) platform to touch the bell. If the player falls into the gap, a death animation is played, which lasts for approximately 1 second. Additionally, it takes approximately 1 second to fall to the bottom of the screen and play the death animation. The level timer counts down during the death animation. The level timer for this task is 30 seconds. It is possible to complete this task 14 times in a single round.

The first set of frametime spike triggers are positioned such that a frametime spike occurs right before the player jumps. If the player jumps before or during a frametime spike, their jump will not go far enough to clear the gap.

The second set of frametime spike triggers are positioned such that a frametime spike occurs right before the player releases the right key to land on the platform.

This task has a precision/deadline/impact score of 3/3/5. The frametime spike delay is 10-207 ms. Figure 9 shows the level layout for this task, which consists of the player spawn position, the frametime spike triggers, and the bell.

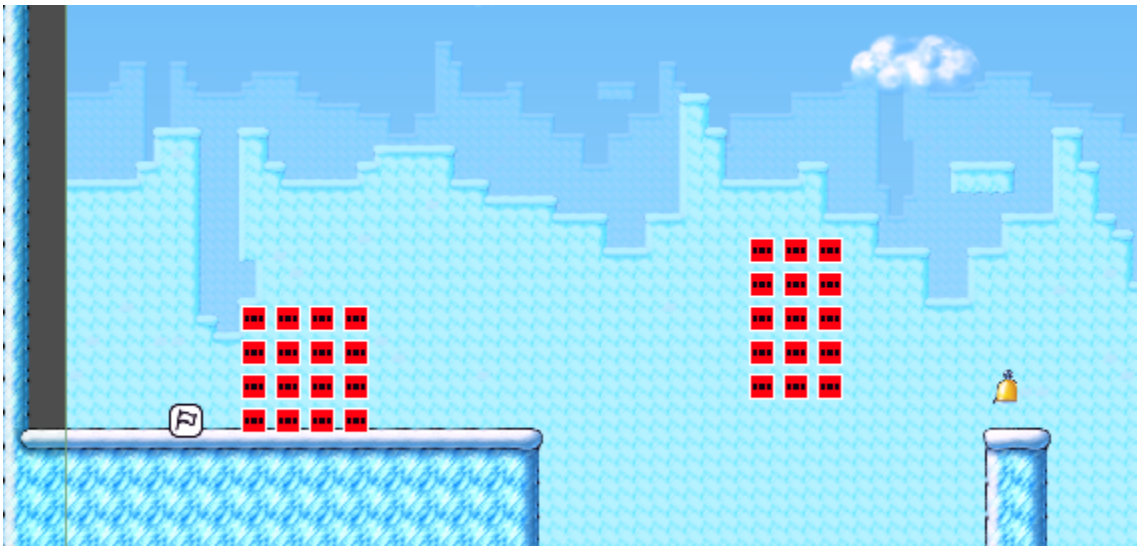


Figure 9: The “Small Gap, Medium Platform” Level.

4.6.2.6 Medium Gap, Small Platform

The goal of this task is to jump over a medium (15-block) gap and touch the bell on the other side. The player must land on a small (1-block) platform to touch the bell. If the player falls into the gap, a death animation is played, which lasts for approximately 1 second. Additionally, it takes approximately 1 second to fall to the bottom of the screen and play the death animation.

The level timer counts down during the death animation. The level timer for this task is 30 seconds. It is possible to complete this task 14 times in a single round.

The first set of frametime spike triggers are positioned such that a frametime spike occurs right before the player jumps. If the player jumps before or during a frametime spike, their jump will not go far enough to clear the gap.

The second set of frametime spike triggers are positioned such that a frametime spike occurs right before the player releases the right key to land on the platform.

This task has a precision/deadline/impact score of 3/4/5. The frametime spike delay is 10-207 ms. Figure 10 shows the level layout for this task, which consists of the player spawn position, the frametime spike triggers, and the bell.

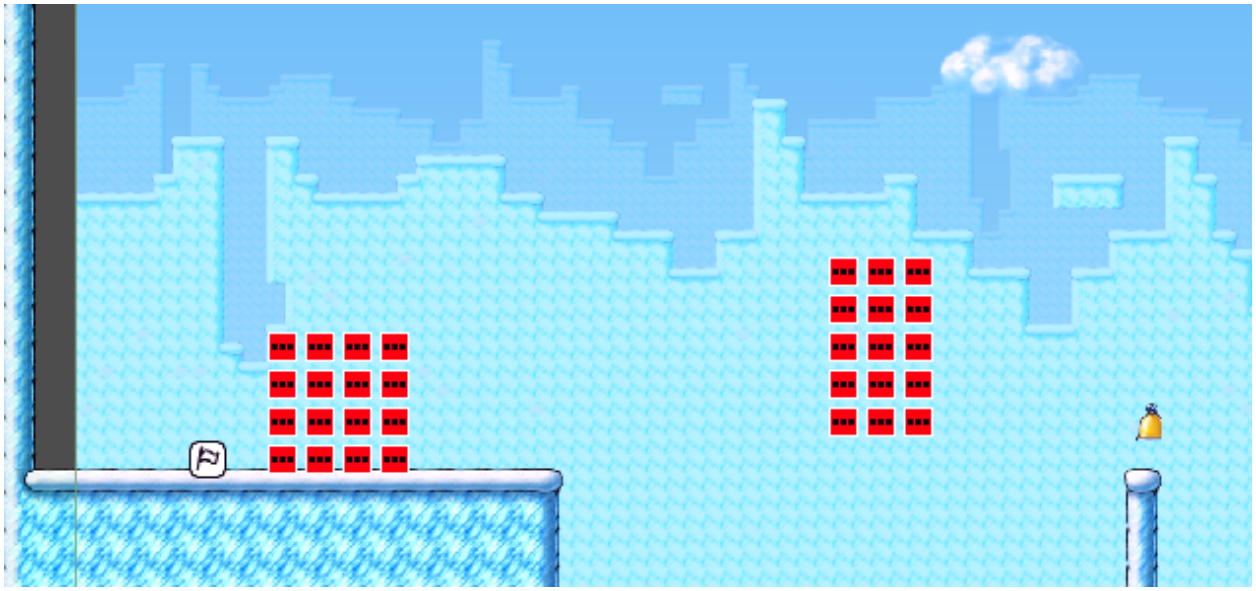


Figure 10: The “Medium Gap, Small Platform” Level.

4.6.2.7 7-Jump

The goal of this task can be split into two sections: first, jumping to the right over the lava; and second, hooking back to the left, under the lava, to touch the bell. If the player touches the lava, a death animation is played, which takes 1-2 seconds depending on the player's y-level when they die (the higher the player is, the longer the death animation takes). The level timer counts down during the death animation. The level timer for this task is 30 seconds. It is possible to complete this task 15 times in a single round.

The first set of frametime spike triggers are positioned such that a frametime spike occurs right before the player jumps. Because of the player spawn position, we cannot ensure that the frametime spike triggers right before the player jumps, but it is very likely to trigger after as players will need speed to clear the lava and the fastest way to get speed is to run forward.

The second set of frametime spike triggers are positioned such that a frametime spike occurs right before the player hooks around to the left.

This task has a precision/deadline/impact score of 4/4/5. The frametime spike delay is 10-69 ms. Figure 11 shows the level layout for this task, which consists of the player spawn position, the frametime spike triggers, the lava position, and the bell.

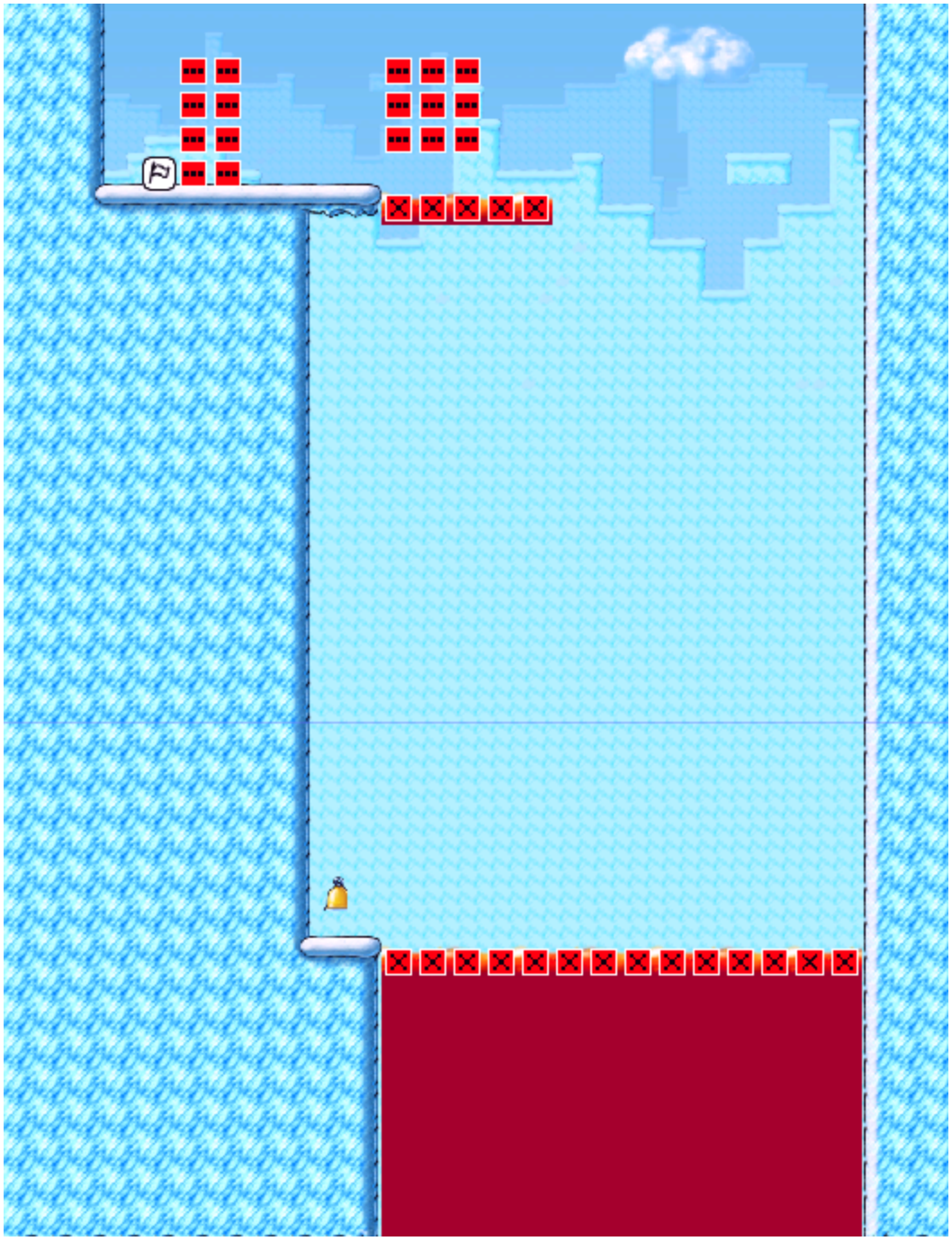


Figure 11: The “7-Jump” Level.

4.6.2.8 Z-Jump

The goal of this task can be split into three sections: first, jumping to the right over the lava; second, hooking back to the left, under the lava; and third, hooking back to the right to touch the bell. If the player touches the lava, a death animation is played, which takes 1-2 seconds depending on the player's y-level when they die (the higher the player is, the longer the death animation takes). The level timer counts down during the death animation. The level timer for this task is 30 seconds. It is possible to complete this task 16 times in a single round.

The first set of frametime spike triggers are positioned such that a frametime spike occurs right before the player jumps. Because of the player spawn position, we cannot ensure that the frametime spike triggers right before the player jumps, but it is very likely to trigger after as players will need speed to clear the lava and the fastest way to get speed is to run forward.

The second set of frametime spike triggers are positioned such that a frametime spike occurs right before the player hooks around to the left.

The third set of frametime spike triggers are positioned such that a frametime spike occurs right before the player hooks back to the right.

This task has a precision/deadline/impact score of 5/5/5. The frametime spike delay is 10-69 ms. Figure 12 shows the level layout for this task, which consists of the player spawn position, the frametime spike triggers, the lava position, and the bell.

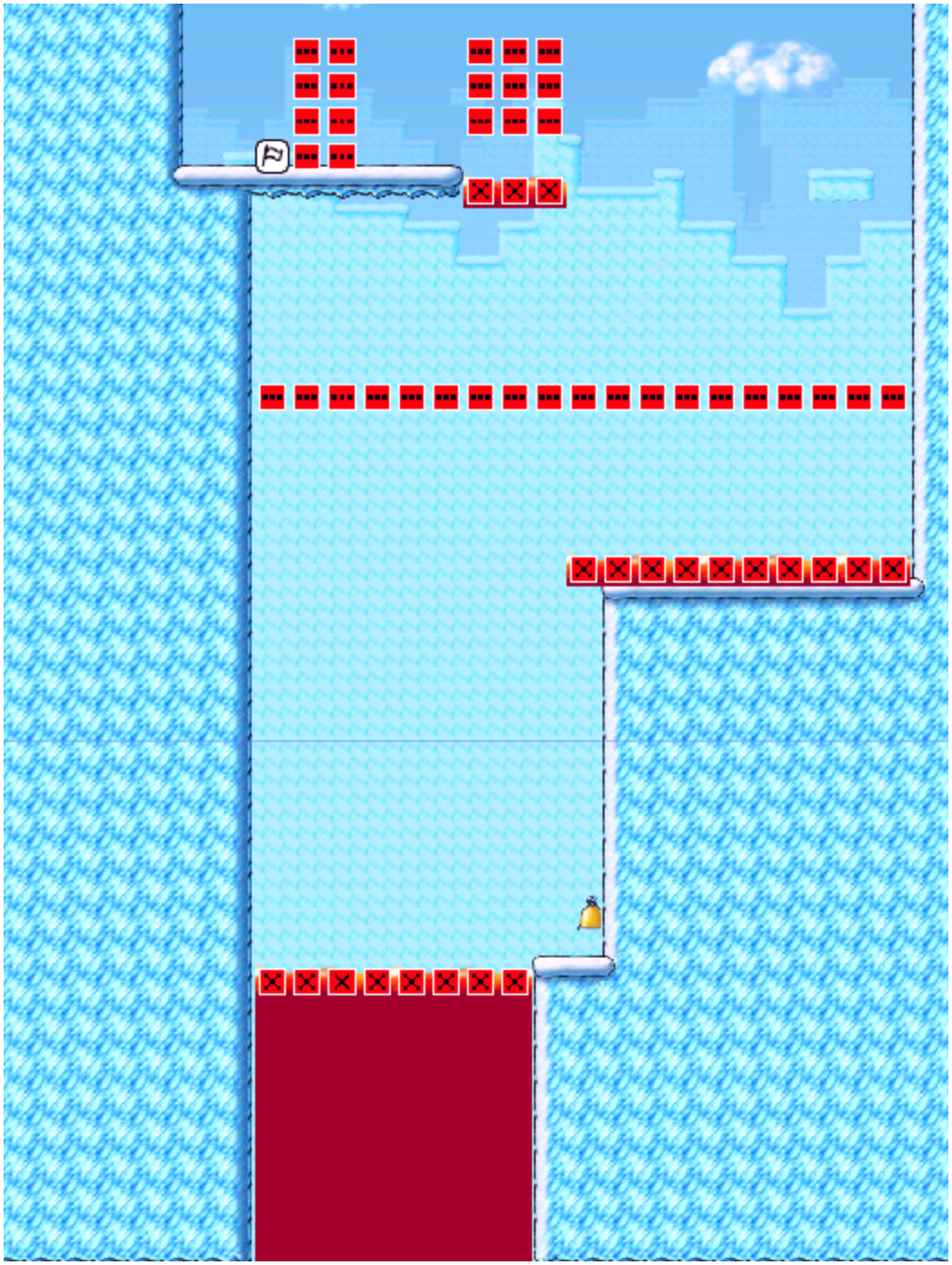


Figure 12: The “Z-Jump” Level.

4.6.3 Creating the Practice Levels

The practice level is designed to incorporate important components of all round levels into itself so that players do not have to learn how the different tasks work in the rounds, which could skew results. The practice level is a continuous stretch, consisting of picking up both types of objects, crossing each gap size, and the two hardest jump types, the 7-jump and the Z-jump. To ensure the players get a chance to try the two hardest jumps, when the practice level timer reaches 40 seconds, if the player has not reached the 7-jump yet, the player will respawn at the 7-jump checkpoint. At 20 seconds left, the same procedure is applied to the Z-jump checkpoint. After the first practice level, which is done with no lag running, a second round of the practice level is run with the maximum amount of lag that will be present in the round levels to ensure players know what both sides of the spectrum will look like. Players are told that the first practice level is the smoothest possible experience and the second practice level is the choppiest possible experience.

4.7 The Playtesting Harness

The playtesting harness handles tasks relating to the administration of the study, including round switching, score calculation, and QoE popups.

4.7.1 User ID

The harness stores the current user ID in a file and increments it by one when a new session is started. This is all done automatically to ensure there is no user error in inputting the user ID.

4.7.2 Round Order

The round order is determined via Latin squares. This allows for an even distribution of round orders across the study.

4.7.3 Player Spawn Positions

Player spawn positions are varied via a uniform random distribution of ±1 block to ensure that participants cannot do the same inputs at the exact same times, which would nullify the effect of frametime spikes.

4.7.4 Background Distractions

Background distractions, such as fish, have been removed to prevent players from getting distracted. Figure 13 shows an example of these distractions.



Figure 13: A Fish in the background of the level.

4.7.5 Popups

The harness controls two categories of popups: the pre-round popup, which is shown at the start of each round, and the post-round popups, which are shown at the end of each round.

4.7.5.1 Pre-Round Popup

The pre-round popup is shown before each round consists of two lines of text. Each round has an objective that varies depending on the round (Table 3). The first line states the objective of the round, and the second line tells the player the objective of the round, as Figure 14 shows:

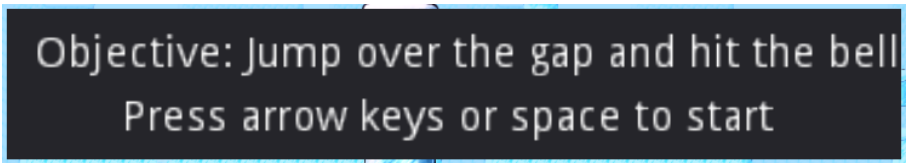


Figure 14: Pre-round popup.

Round Type	Objective Text
Practice	Complete the level.
Small Gap, Medium Platform	Jump over the gap and hit the bell.
Medium Gap, Small Platform	Jump over the gap and hit the bell.
Large Gap, Large Platform	Jump over the gap and hit the bell.
Collect Item: Moving	Collect the bell.
Collect Item: Bouncing	Collect the bell.
Squish Enemy	Jump on top of the enemy.
7-Jump	Jump down to hit the bell.
Z-Jump	Avoid the lava and hit the bell.

Table 3: The Different Objectives for Each Level.

Extra text is shown during the practice rounds. In addition to the text that is shown on the regular pre-round popup, this popup also has four additional lines of text. The first line informs the player that they are currently in a practice round, the second and third lines tell the player the controls, and the fourth line tells the player whether the current practice round is the smoothest or the choppiest as Figure 15 shows:

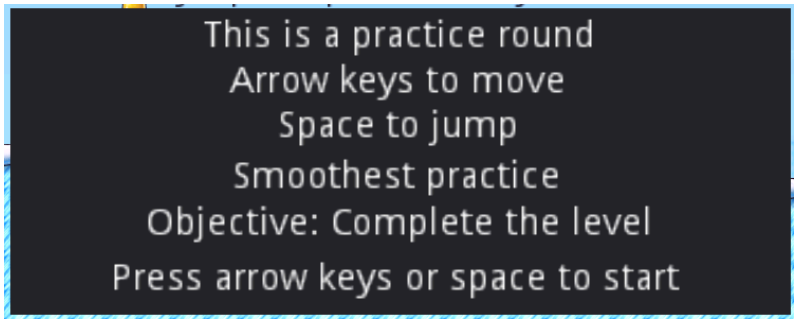


Figure 15: The practice level pre-round popup. This is shown exclusively during practice rounds.

4.7.5.2 Post-Round Popups

Two post-round popups appear after the timer reaches zero. The first popup asks the player to rate the visual smoothness of the round (see Figure 16). After that popup is closed, a second popup appears that asks the player if the smoothness of the round was acceptable (see Figure 17). In both cases, the player must interact with the popup before it can be closed.



Figure 16: Post-round QoE score popup. The first post-round popup, before (left) and after (right) the player interacts with the slider. The slider ranges from 1 (low) to 5 (high), with increments of 0.1.

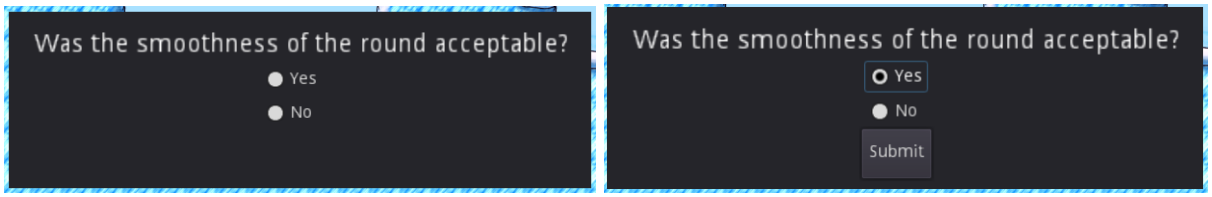


Figure 17: Post-round QoE acceptability popup. The second post-round popup, before (left) and after (right) the player interacts with the radio buttons.

4.7.6 Player Score

The score is incremented by 100 every time the player completes a round's objective. No penalty is given for failure because it already incurs a time penalty: the player must wait for the

death animation to complete before resuming play. The score increments normally in practice levels and is reset to zero when the rounds start. Figure 18 shows the player score GUI element.

4.7.8 Round Counter

The round counter is positioned directly under the player score. It shows the current round the player is on and the total number of rounds in the session. The round counter is not visible in practice levels. Figure 18 shows the round counter GUI element.



Figure 18: Player info GUI. The GUI element shows the player's score and the current round.

4.8 Logging System

Four different log types are written during the playtesting session: a Frame Log, Event Log, Quality of Experience (QoE) Log, and Summary Log. These logs are for data collection of a variety of different metrics and aggregates that will inform the results of the study. At the start of a playtesting session, the user is assigned an arbitrary ID number that is then associated with a folder containing a .csv log for each of the four types of logs.

4.8.1 Log Types

A Frame Log stores a line of player data every frame that the engine runs. Data from this log includes: Time, PlayerID, DeltaFrames, Current Level, Game State, Number of Lives,

Deaths, X-Position, Y-Position, X-Velocity, Y-Velocity, and Game Tick Rate. This data is helpful when comparing against data from other logs and to ensure there are no discrepancies between events or game performance. This data in this log is locally stored and then written to a file once the final question in the QoE survey is complete and the harness moves to the next round.

An Event Log stores a line of player data every time an event occurs in the game. Events include state changes, player input to the keyboard, frametime spikes with duration, task success, and task failures. Data from this log includes: Time, PlayerID, DeltaFrames, Expected Lag, Current Level, Game State, Number of Lives, Deaths, X-Position, Y-Position, X-Velocity, Y-Velocity, and an Event. This data is useful when keeping track of a player's number of successes, failures, and inputs. Also the Event Log helps verify that the duration of a frametime spike that is thrown is consistent with its expected duration. This data in this log is locally stored and then written to a file once the final question in the QoE survey is complete and the harness moves to the next round.

A QoE Log stores a line of player data for every QoE question occurring after a practice or playtesting round. A QoE question is either an answer to the 1-5 scale of the visual quality of a round or a yes/no answer regarding the acceptability of a round. Data from this log includes: PlayerID, Time, Current Level, and an Event. This data is meaningful because it records the quality of experience of the player for each round to conclude how QoE is impacted by frame latency. This data in this log is locally stored and then written to a file once the final question in the QoE survey is complete and the harness moves to the next round.

A Summary Log stores data and calculates aggregate data for each round of gameplay (round summary) as well as aggregates for all of the round data combined (game summary). The round summary data includes: Average FPS, Level Time, Total Frames, Total Events, Total

Deaths, Total Successes, Total Failures, Total Inputs, Total Jumps, Total Left Inputs, Total Right Inputs, Round Acceptability (yes/no), and the Round QoE Score. The game summary data includes: Total Frames, Total Events, Total QoE Entries, Average QoE Score, and the Number of Acceptable Rounds. The data is meaningful as aggregates help understand the crucial data within any given round and for the playtesting session as a whole. This data in this log is locally stored and then written to a file once the game is closed and the playtesting session is over.

5. Results

5.1 Outlier Analysis

We examined data from this study for indicators that warrant the removal of any play tester or level from data analysis. We checked each player's performance, QoE score, and activity against the standard deviation of the data set, and flagged any player that fell outside of these bounds. This analysis resulted in flagging three participants. Two of these participants, IDs 349 and 352, had overlapping categories. After further investigation of the flagged players through cross-referencing other round data and the demographics survey, these flagged participants are very experienced players in 2D platformers, which explains their performance and they remain valuable data points in this study.

We investigated both success and activity on rounds. We flagged three rounds: squish bouncing enemy, large gap jump with large platform, and small gap jump with medium platform. All three of these levels were flagged in the activity category. These levels are all levels where a player would be expected to be cautious or dying often, which explains the decreased activity for these levels, and they remain valuable data points for this study.

We also investigated individual player rounds. We checked every round against the standard deviation in both the number of successes and activity in rounds of the same type by different players. This analysis raised 82 flagged player rounds in the activity category. After removing the levels already flagged for low activity, only 20 flagged rounds remained, however, the activity scores suggest that each player still gave these levels a fair try, and these flagged player rounds remain valuable data points that do not warrant removal. In short, we kept all participants’ data for analysis.

5.2 Data Analysis

5.2.1 Performance

We analyzed the data collected in this study to gain insight into how player performance was impacted by the different frametime spike sizes and tasks. One of the metrics used to help quantify a player's performance is success rate. We calculated the *actual* success rate based on the number of total successes over attempts made by the player. We computed the *practical* success rate by taking a given player's number of successes in a round and dividing it by the total number of possible successes in a given round. For example, if a player completed the Z-Jump task four times in a round out of eight attempts, their *actual* success rate would be 50% because they succeeded in half of the attempts. However, their *practical* success rate would be 25% because the Z-Jump can be completed a maximum of 16 times in a single round. Figure 19 shows a negative correlation between frametime spike time and practical success rate over all rounds. As frametime spike time increased, player success rate decreased.

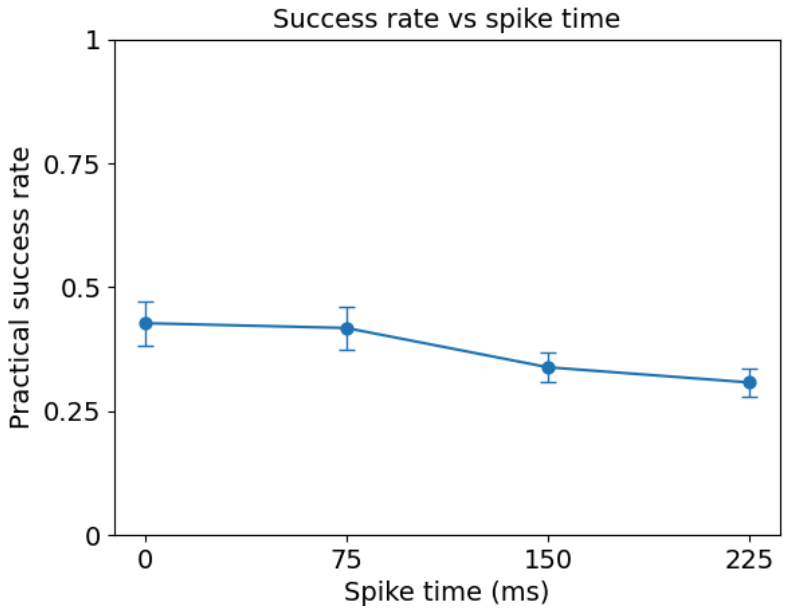


Figure 19: Practical Success Rate vs. Frametime Spike Size (ms). The error bars show the 95% confidence intervals.

Another area of interest is the potential implications that a given task has on a player's success rate over all frametime spike sizes (Figure 20). The Collect Moving Collectible task had the highest success rate, whereas the Z-Jump had the lowest success rate. This aligns with the way these levels were designed as the former was designed to be the easiest task and the latter was designed to be the hardest task.

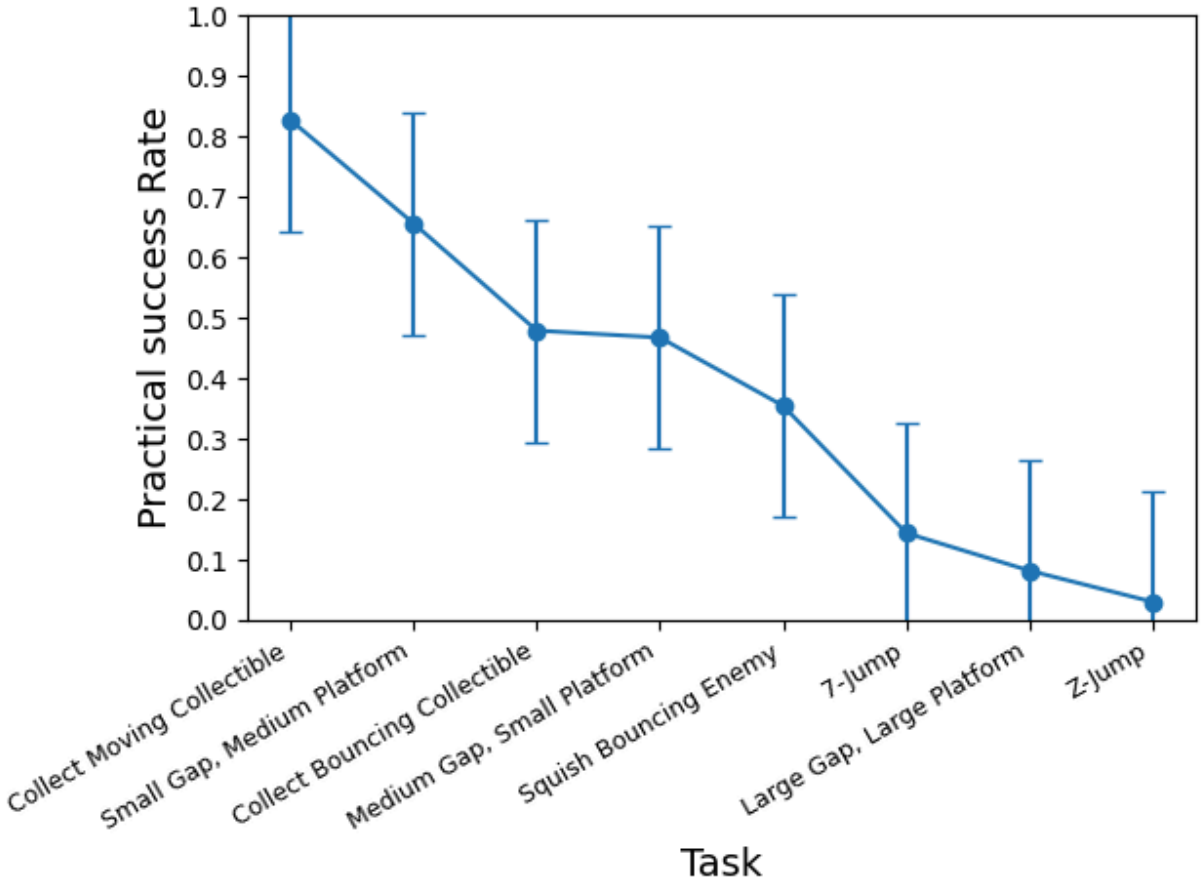


Figure 20: Practical Success Rate vs. Task. The error bars show the 95% confidence intervals.

This correlation supports our hypothesis that player performance decreases as latency increases. However, when breaking down tasks into their "task groupings" it is clear that some groupings are more impacted than others by frametime spike size (Figure 21 & Figure 23). As shown, the Collectible and Squish Enemy task groupings do not show a significant correlation between the different spike sizes. However, the Jump Over Gap and Special Jump task groupings do display a negative correlation between frametime spike time and success rate. The slopes of the relationship between practical success rate and spike size show a significant decline with Jump Over Gap grouping and a gradual decline with the Collectible and Special Jump groupings

(Figure 22). This could be due to the more predictable nature of the collectible and enemies tasks, which require less individual time-sensitive inputs when compared to jumping tasks.

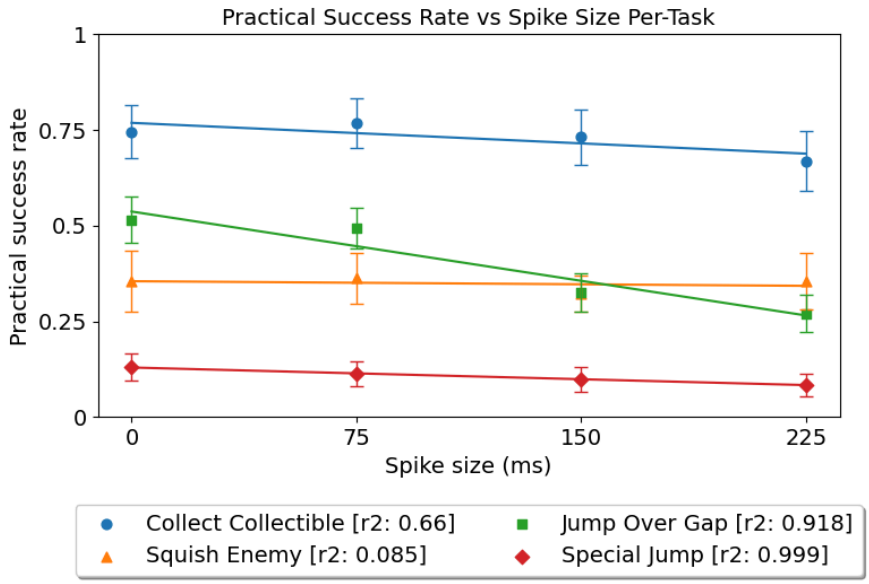


Figure 21: Practical Success Rate vs. Spike Time Per-Task Grouping. The error bars show the 95% confidence intervals.

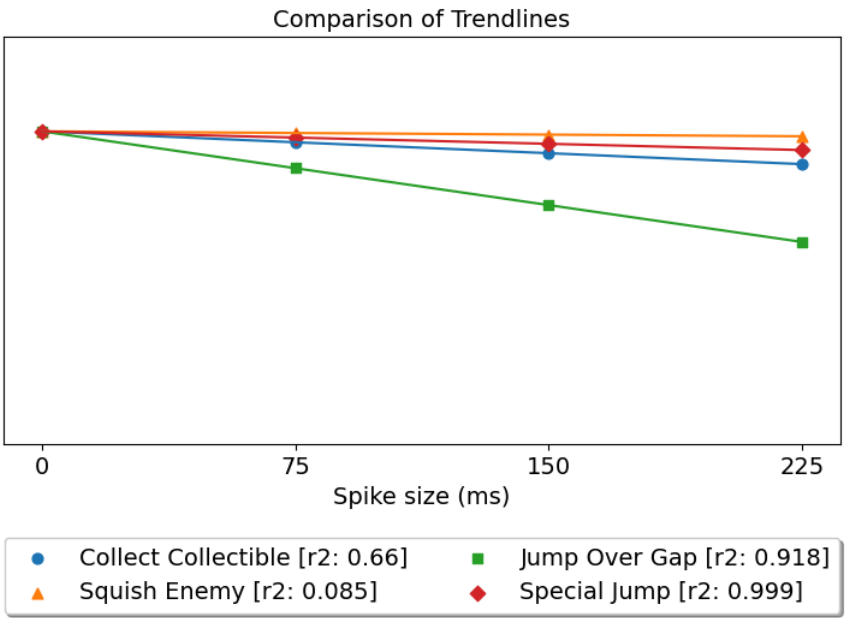


Figure 22: Comparison of Trendlines in Figure 21.

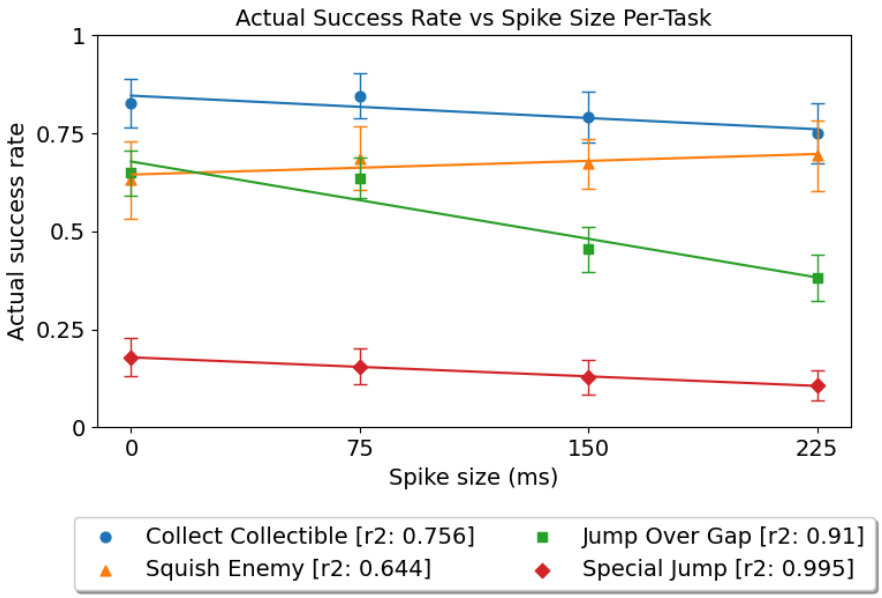


Figure 23: Actual Success Rate vs. Spike Time Per-Task Grouping. The error bars show the 95% confidence interval.

We analyze the *actual* success rate with the *practical* success rate to derive an understanding of their comparative relationships versus frametime spike size. When considering the task groupings, in all cases, the actual success rate was higher than the practical success rate (Figure 24). There is a significant difference in the actual and practical success rate for the Squish Enemy, whereas with the other remaining groupings, both lines are very close. Figure 25 demonstrates that when computed by individual tasks, this remains true. In cases such as Squish Bouncing enemy and Small Gap, Medium Platform, there is a notable difference in actual vs. practical success rate which suggests that practical success rate is not an accurate measure of actual success.

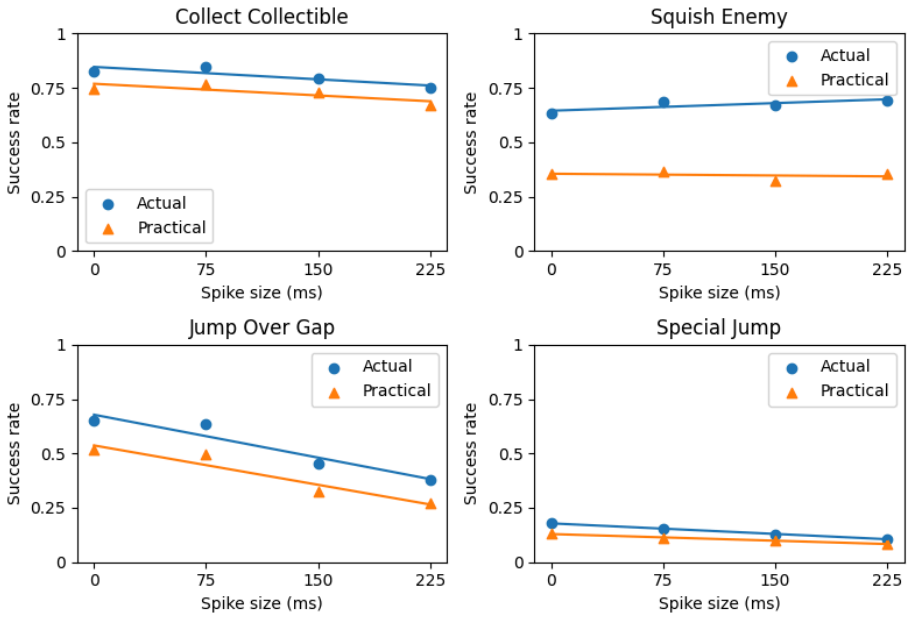


Figure 24: Actual and Practical Success Rate vs. Spike Time Per-Task Grouping.

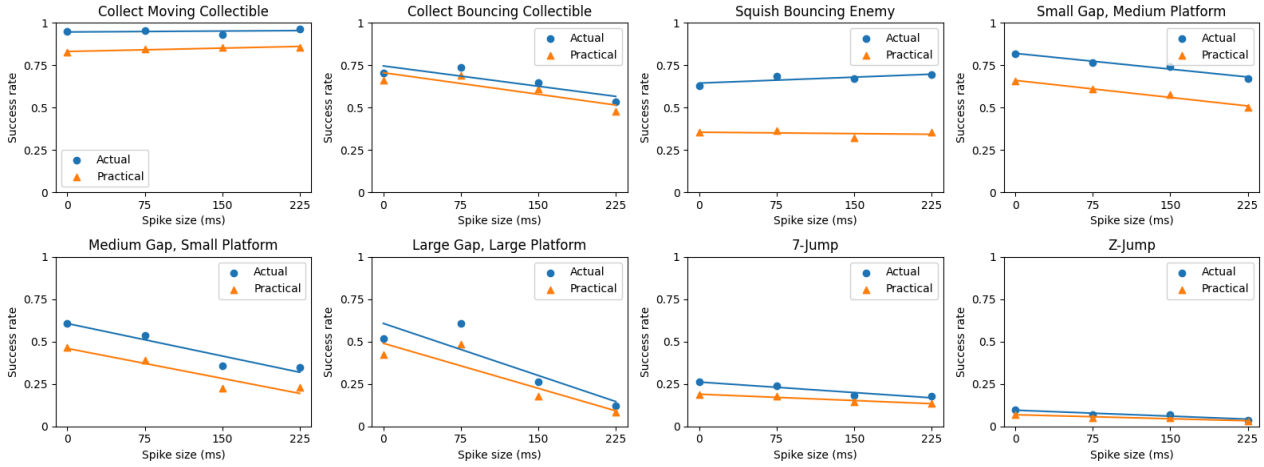


Figure 25: Actual and Practical Success Rate vs. Spike Time Per-Task.

We also analyzed each task to determine if there was any correlation between success rate and frametime spike size. To look more in depth at each task, the Large Gap, Large Platform had the clearest negative decline in practical success when in relation to frametime spike size.

Instead, tasks like Collect Moving Collectible and Squish Bouncing Enemy seemed to have a negligible decline (Figure 26 & Figure 27).

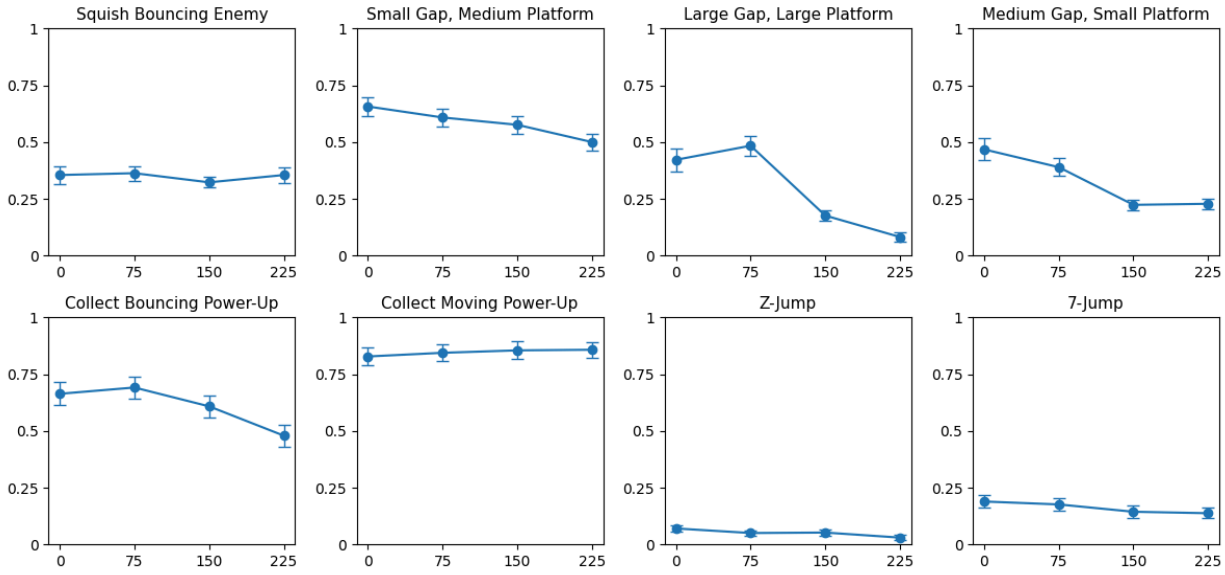


Figure 26: Practical Success Rate vs. Frametime Spike Size (ms) Per-Task. Each graph represents an individual task as titled above the respective graph. The error bars show the 95% confidence intervals.

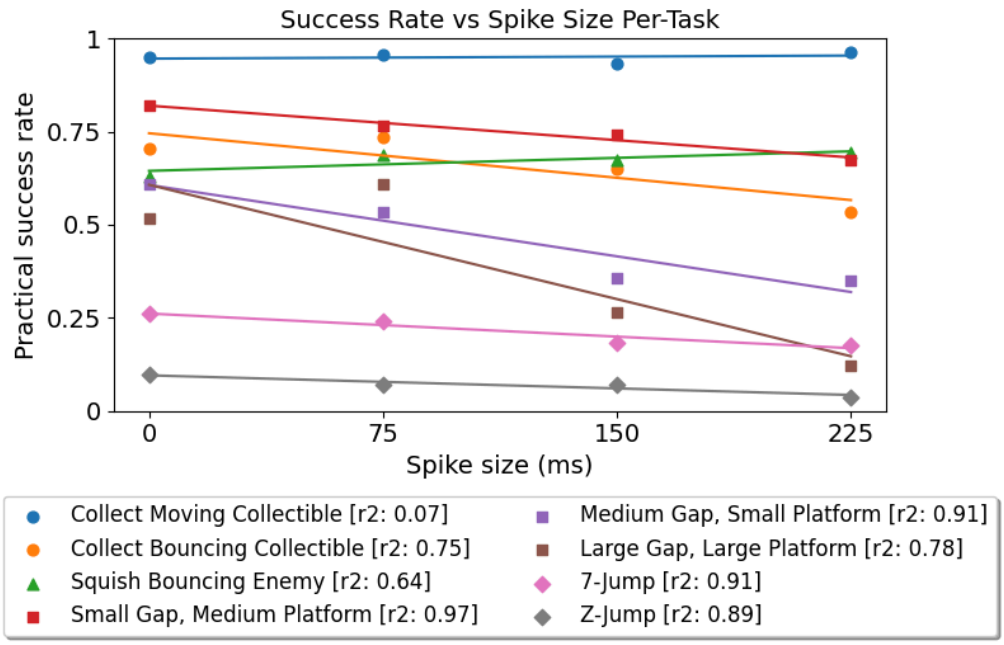


Figure 27: Comparison of Practical Success Rate vs. Spike Time Per-Task (ms).

5.2.2 Quality of Experience

The other major comparison lies in a player's quality of experience (QoE) as it relates to the different frametime spike sizes and tasks. These results give us further insight into the way frametime variation impacts the player's experience and the thresholds at which a player finds the experience unacceptable. They also uncover which tasks are most sensitive to QoE score variations due to different frametime spike sizes. Figure 28 shows a definite decline in the QoE score (on a 1-5 scale) as the frametime spike size increases.

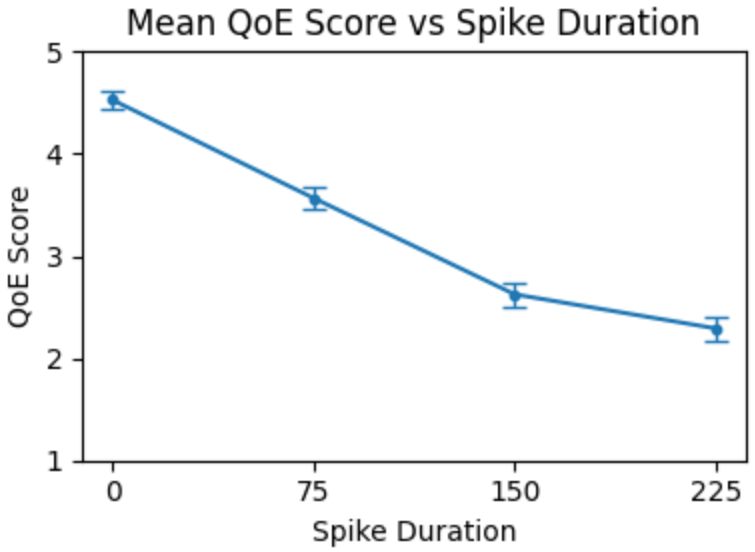


Figure 28: Mean QoE Score vs. Frametime Spike Size (ms). The error bars show the 95% confidence interval. The data are taken over all tasks and frametime spike sizes.

This negative correlation is further supported when each of the four "task groupings" is analyzed to see how frametime spike size impacts QoE score, as Figure 29 shows. The data demonstrate a very similar negative correlation for each of the four "task grouping". It appears that the Special Jumps grouping is slightly more sensitive as frametime spike increases.

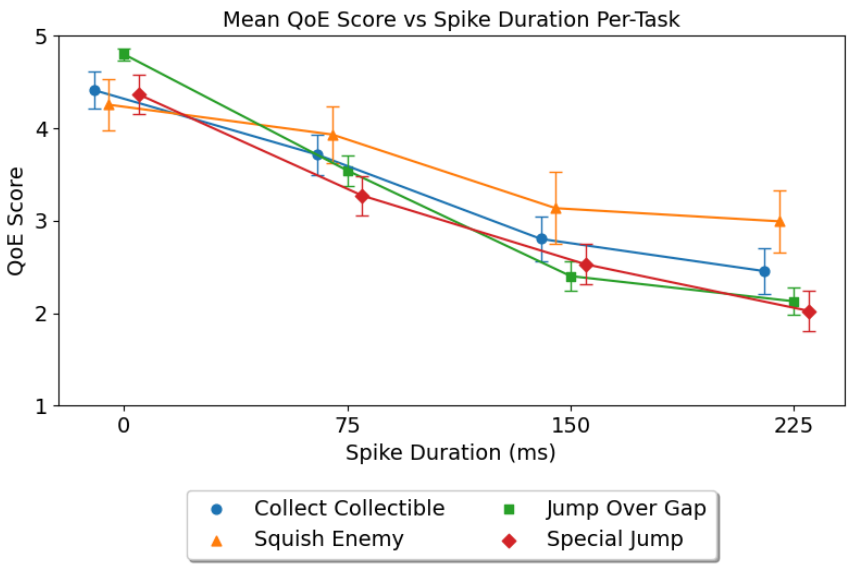


Figure 29: QoE Score vs. Frametime Spike Size (ms) Per-Task Grouping. The error bars show the 95% confidence interval.

Analyzing the distribution of QoE scores for each of the four frametime spike sizes is also valuable, and Figure 30 shows this. As expected, the 0 ms frametime spike size features a very left-skewed distribution with the 75 ms distribution being less dramatically left-skewed. When analyzing the QoE score distribution for the 150 ms and 225 ms frametime spike sizes, the former is slightly skewed-right with the latter being more notably skewed-right. Figure 30, Figure 31, and Figure 32 break this relationship down into the individual tasks. These distributions support the claim that as frametime spikes increase, QoE scores decrease.

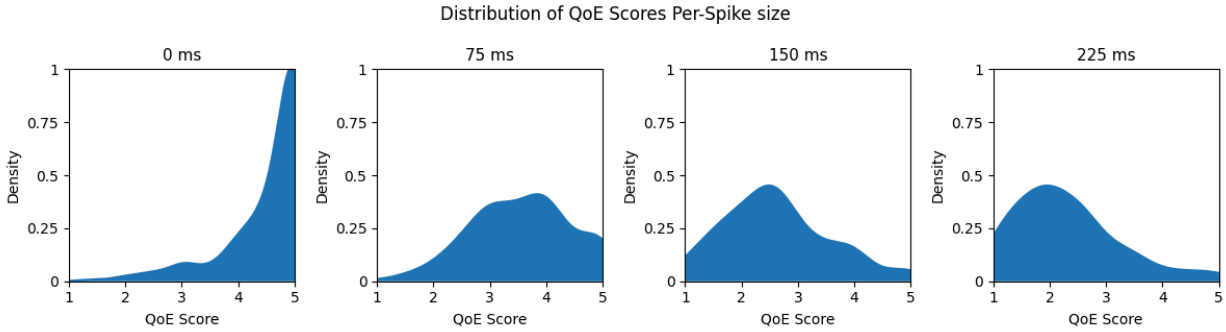


Figure 30: Distribution of QoE Scores Per-Spike Size. These graphs represent the distribution of the total QoE scores for all participants over all tasks and frametime spike sizes.

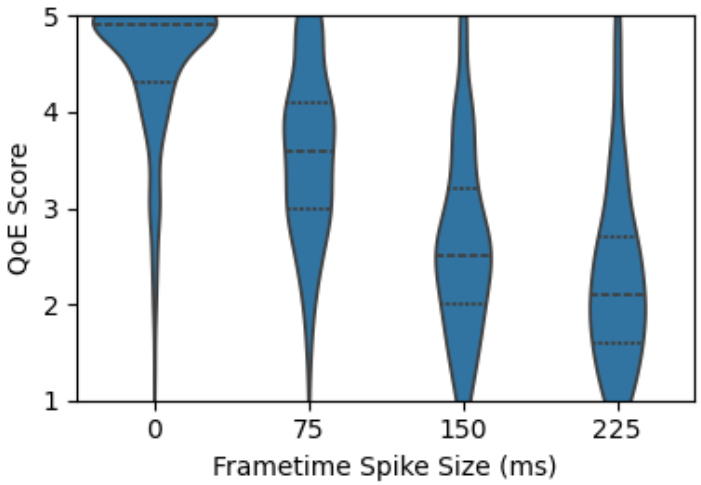


Figure 31: Violin Plot QoE Score vs. Frametime Spike Size (ms).

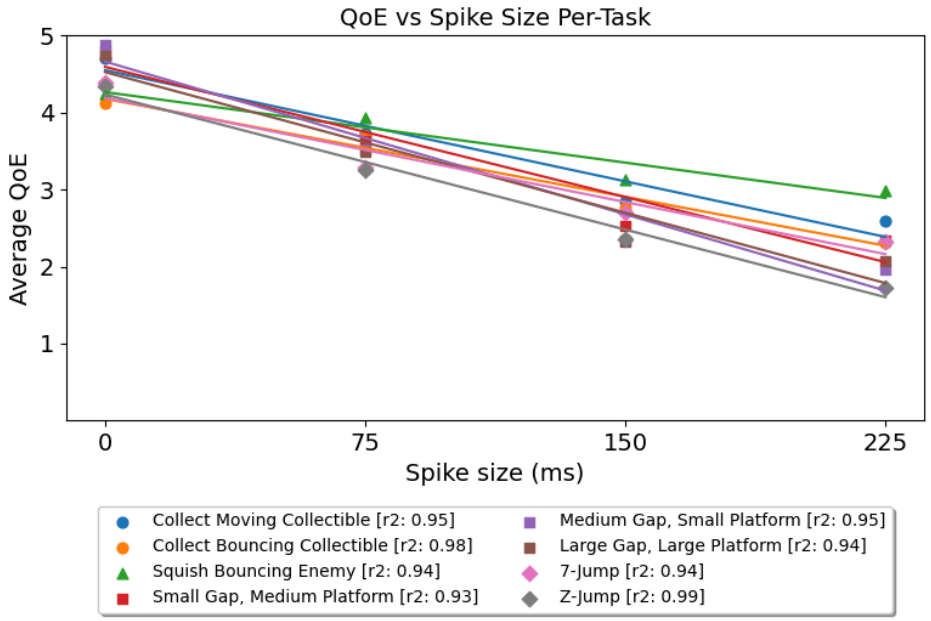


Figure 32: Average QoE Score vs Spike Time.

The average QoE score for each task provides insight into how the individual tasks are scored for each spike size (Figure 33). The Squish Bouncing Enemy task has the smallest decline over time while the Z-Jump task had the most significant change with the increase of frametime spike size. Investigating the overall distribution of QoE scores is also important for understanding how participants rated their playtesting experience, and Figure 34 shows our data. The graph is bimodal, with peaks around QoE scores of 2.6 and 5.0. This means that participants most often rated rounds' QoE at these. We also analyzed the distribution of mean QoE scores. Figure 35 shows a left-skewed distribution of QoE scores meaning that more often than not, a participant's average score over all rounds was above average.

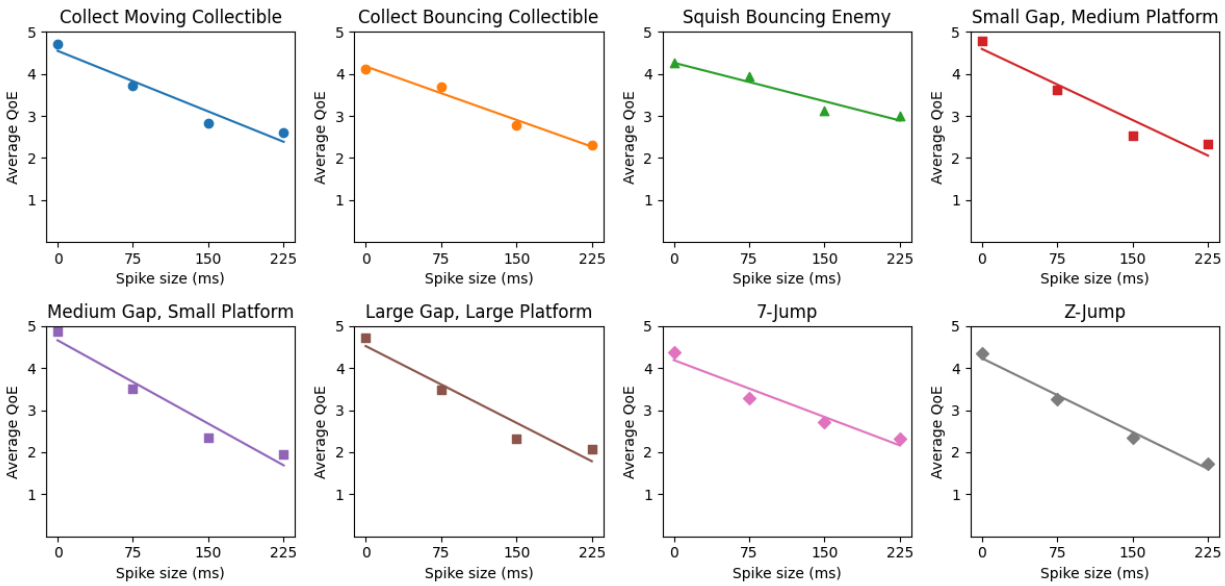


Figure 33: Average QoE vs Spike Time Per-Task. This graph includes all participants over all rounds.

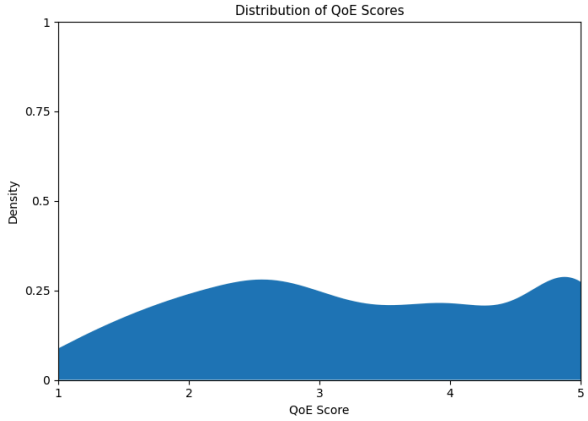


Figure 34: Distribution of QoE Scores. This graph represents the distribution of all QoE scores for all players over all tasks and frametime spike sizes.

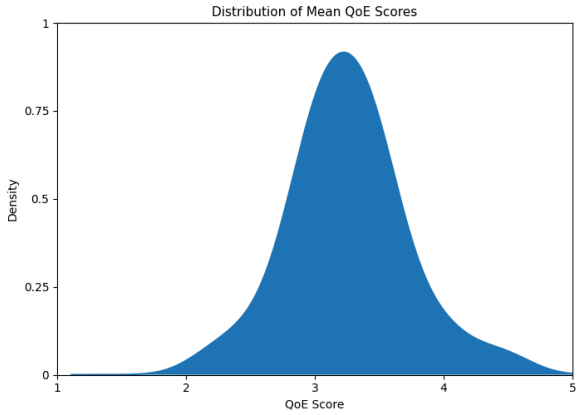


Figure 35: Distribution of Mean QoE Scores. This graph represents the distribution of all participants' mean QoE scores over all tasks and frametime spike sizes.

To look deeper into the QoE score from our data, it is useful to look at the distribution of QoE scores per-user (Figure 36). This helps contextualize the biases a given user may have toward scoring the rounds.

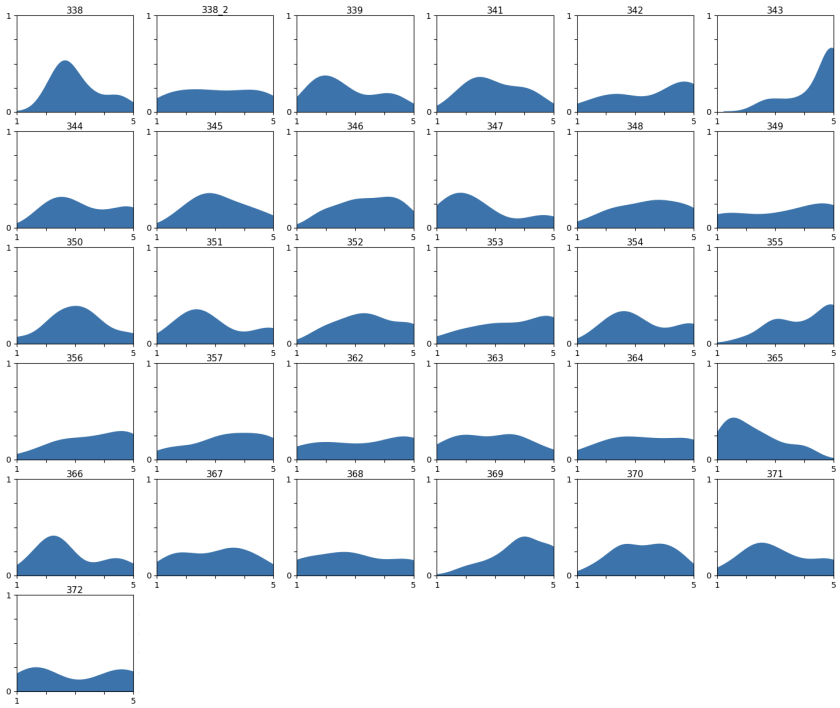


Figure 36: Distribution of QoE Scores Per-User. This graph represents the distribution of each participant's QoE score over all tasks and frametime spike sizes.

Insight can be found when investigating the threshold of QoE score that warrants a participant to consider a given round acceptable. The data suggests, as expected, that as the QoE score increases, the rate at which a round is deemed acceptable also increases (Figure 37). Based on our findings, it can be seen that the point at which a level is more likely than not to be deemed acceptable lies around a QoE score of 3. This aligns well with the methodology to have a QoE score of three to represent an average experience.

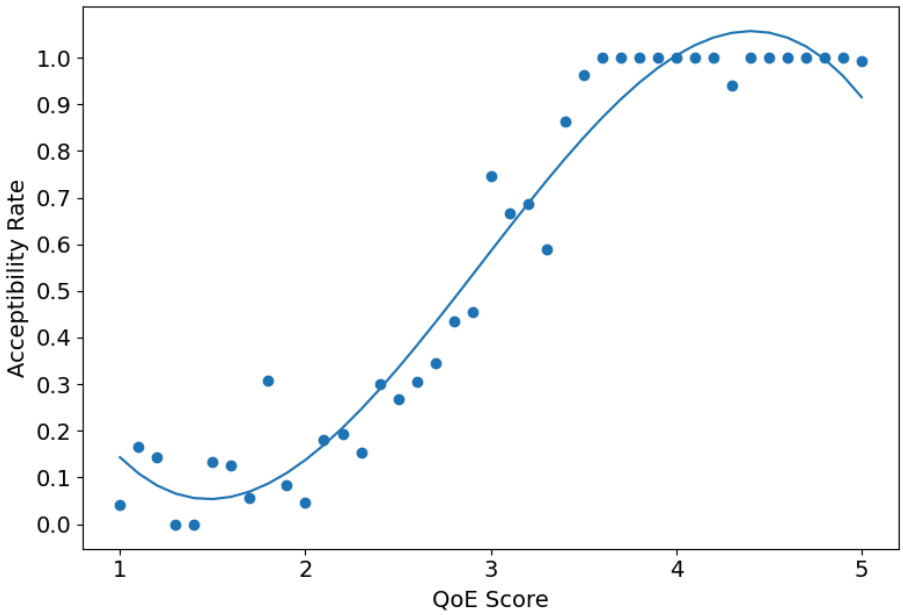


Figure 37: Acceptability Rate vs. QoE Score. This graph represents the acceptability rate, or the rate at which a round was deemed acceptable based on the accompanying QoE score for the round. This data was taken over all participants, tasks, and frametime spike sizes.

5.3 Precision/Deadline/Impact

When applying the precision/deadline/impact rubric to our data, we can investigate relationships between each category to player performance and quality of experience. This will provide insight into whether the precision deadline impact is a meaningful projection of a task's difficulty.

5.3.1 Effectiveness at Predicting Player Performance

Analyzing player performance against each of the categories shows a clear negative correlation between average success rate and precision, deadline, and impact score (Figure 38, Figure 39, & Figure 40). This suggests that the precision/deadline/impact rubric is an effective indicator of a task's difficulty.

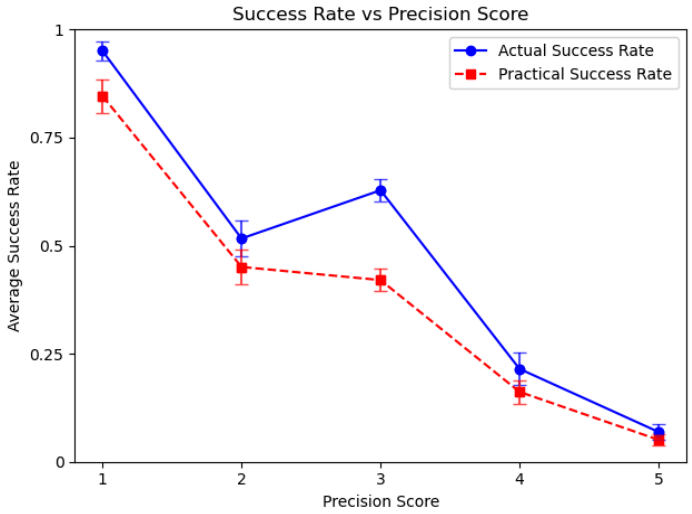


Figure 38: Average Practical and Actual Success Rate vs. Precision Score. The error bars show the 95% confidence interval.

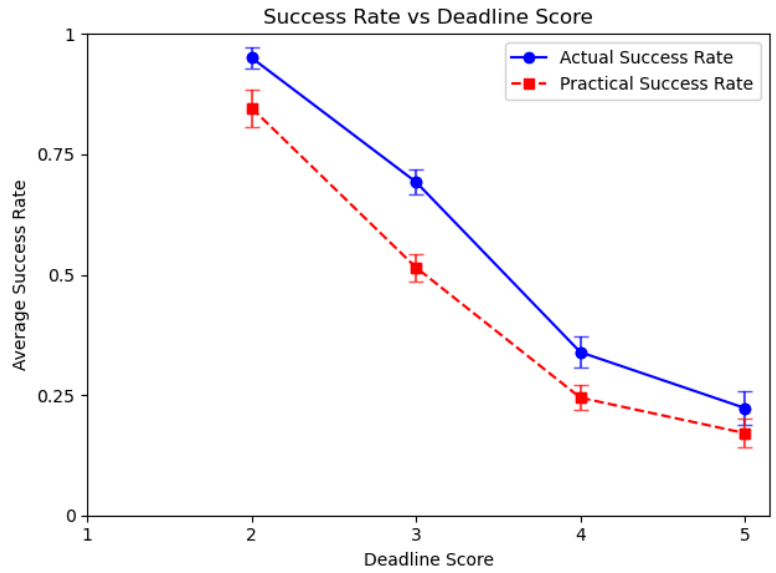


Figure 39: Average Practical and Actual Success Rate vs. Deadline Score. The error bars show the 95% confidence interval.

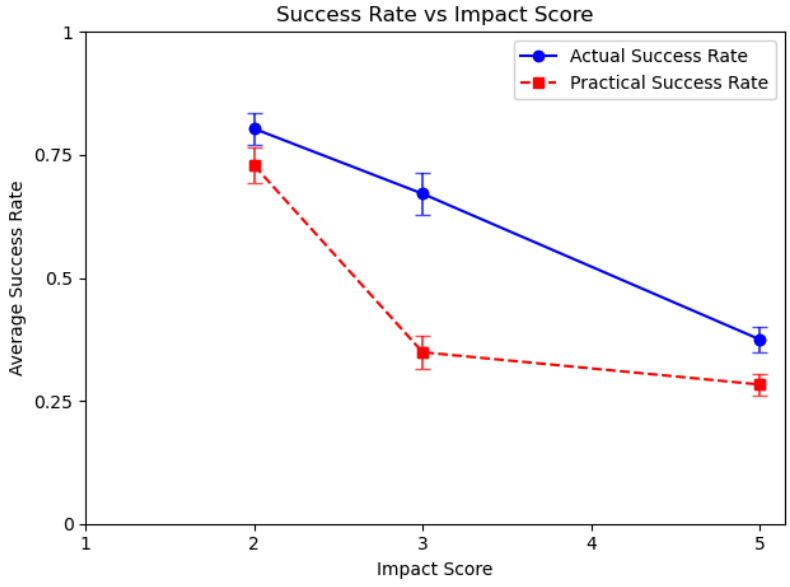


Figure 40: Average Practical and Actual Success Rate vs. Impact Score. The error bars show the 95% confidence interval.

When comparing all scores for both average actual success rate and average practical success rate, there are very similar trends (Figure 41 & Figure 42). However, in many cases, high precision scored tasks are shown to affect player success most significantly. Whereas, in many cases the deadline and impact scores often tend to be similarly predictive of the player performance. It is worth considering the single case where there is not a negative correlation between precision/deadline/impact score and average success rate in Figure 42. This could be due to a misclassification of the two levels that have precision score of 3: “Small Gap, Medium Platform” and “Medium Gap, Small Platform.”

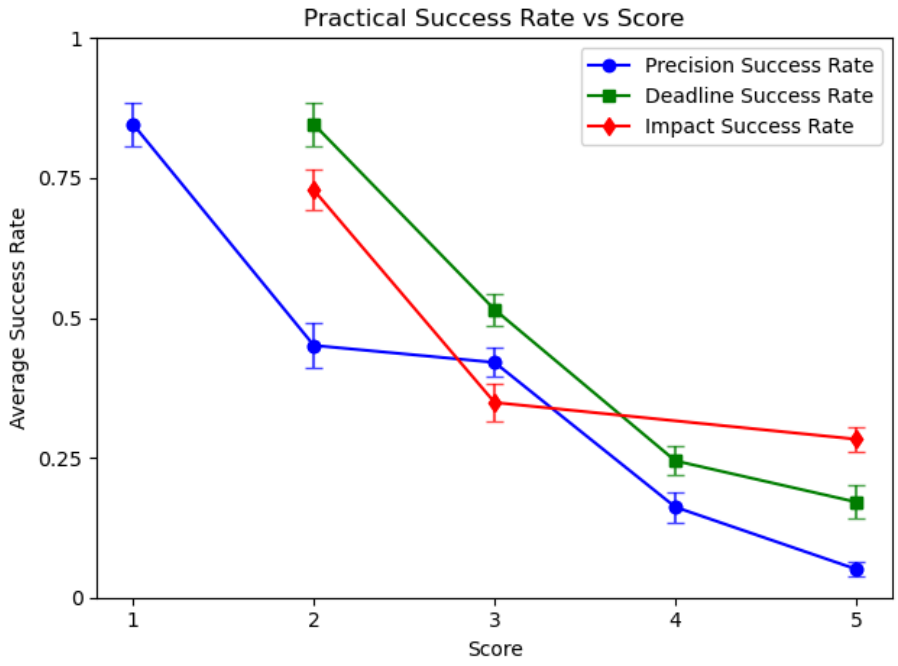


Figure 41: Average Practical Success Rate vs. Precision/Deadline/Impact Score. The error bars show the 95% confidence interval.

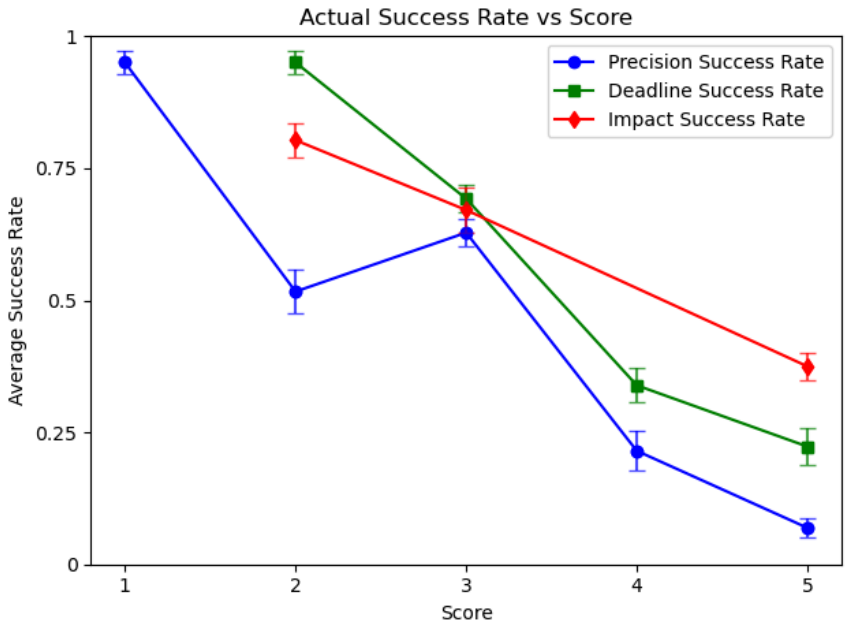


Figure 42: Average Actual Success Rate vs. Precision/Deadline/Impact Score. The error bars show the 95% confidence interval.

5.3.2 Effectiveness at Predicting QoE Scores

Figure 43, Figure 44, & Figure 45 suggest that the precision/deadline/impact score of a task is an unreliable predictor of a player's QoE score.

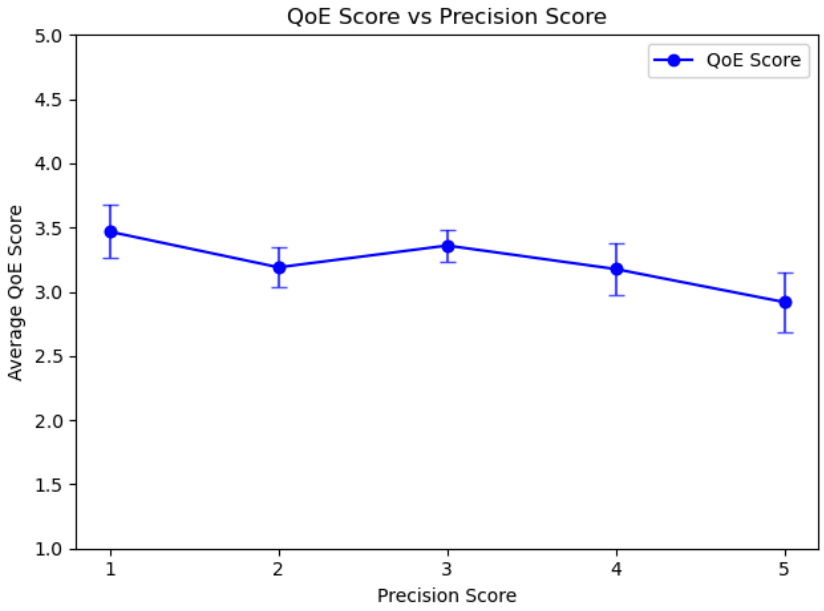


Figure 43: Average QoE Score vs. Precision Score. The error bars show the 95% confidence interval.

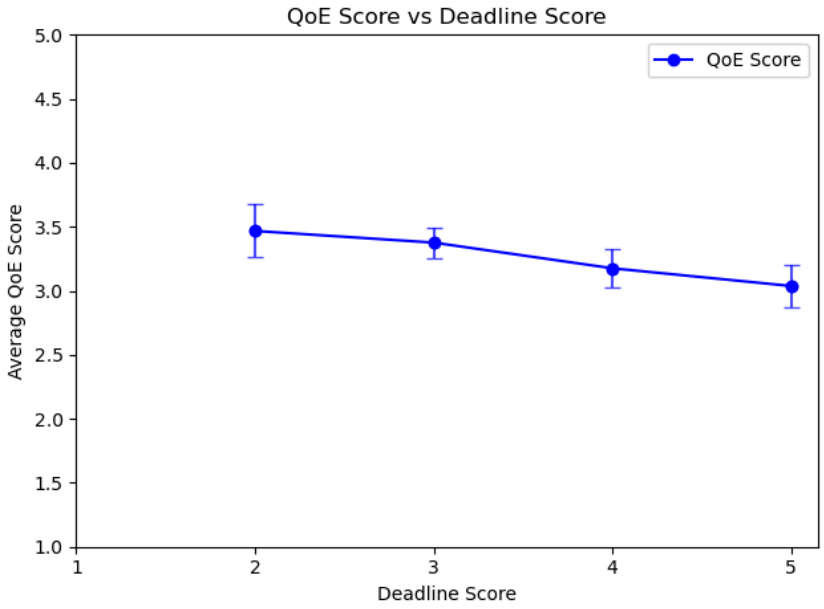


Figure 44: Average QoE Score vs. Deadline Score. The error bars show the 95% confidence interval.

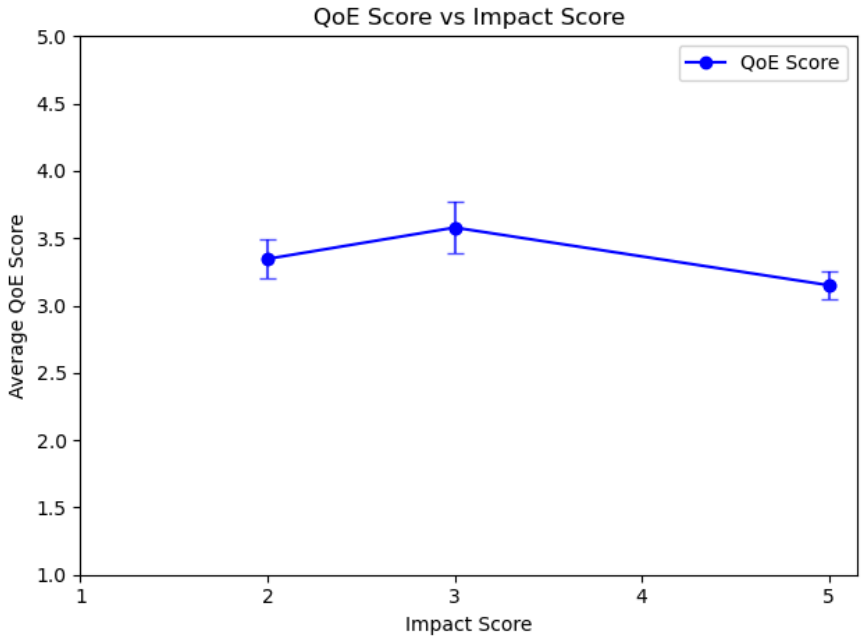


Figure 45: Average QoE Score vs. Impact Score. The error bars show the 95% confidence interval.

When comparing each category together in relation to average QoE score, these inconclusive trends become more apparent (Figure 46). The data show a possibility that all categories could have a slight negative correlation, all data points are within the margin of error.

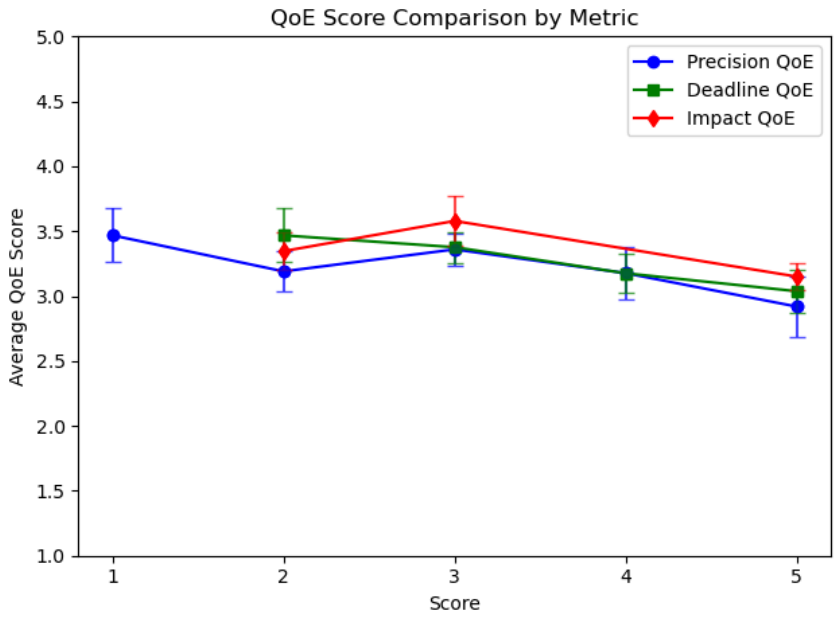


Figure 46: QoE Score vs. Precision/Deadline/Impact Score. The error bars show the 95% confidence interval.

5.3.3 Effectiveness at Predicting Frametime Spike Sensitivity

Figure 47, Figure 48, and Figure 49 show the actual success rate compared to precision for each frametime spike duration, while Figure 50, Figure 51, and Figure 52 show the practical success rate. Figure 47 and Figure 50, for precision 2, and Figure 48 and Figure 51, for deadline 5, show a clear difference in player performance compared to frametime spike duration. This is most likely due to the "Jump over Large Gap, with Large Landing Platform" task, which has a difficulty rating of 2/5/5. All other data points show little to no difference in player performance between frametime spike durations. Excluding the "Jump over Large Gap, with Large Landing Platform" task, the precision/deadline/impact rating of a task is not an effective predictor of a task's sensitivity to frametime spikes.

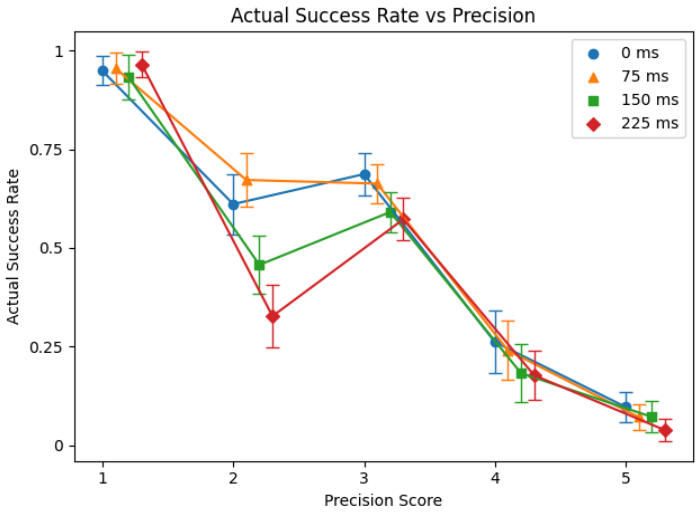


Figure 47: Actual Success Rate vs Precision Per-Frametime Spike Duration. The error bars show the 95% confidence interval.

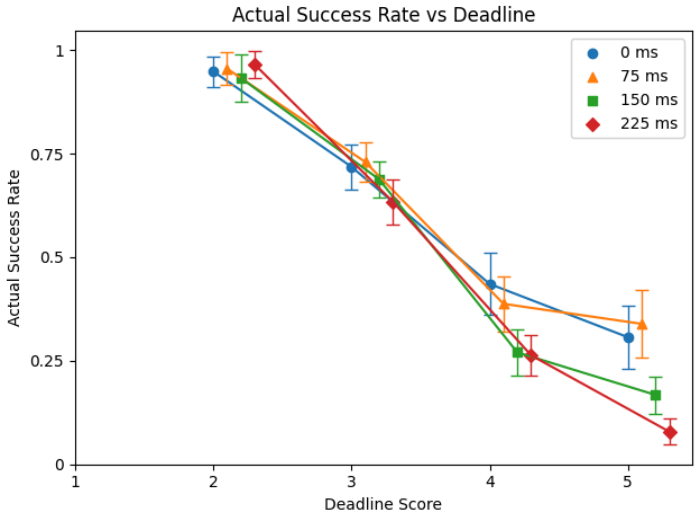


Figure 48: Actual Success Rate vs Deadline Per-Frametime Spike Duration. The error bars show the 95% confidence interval.

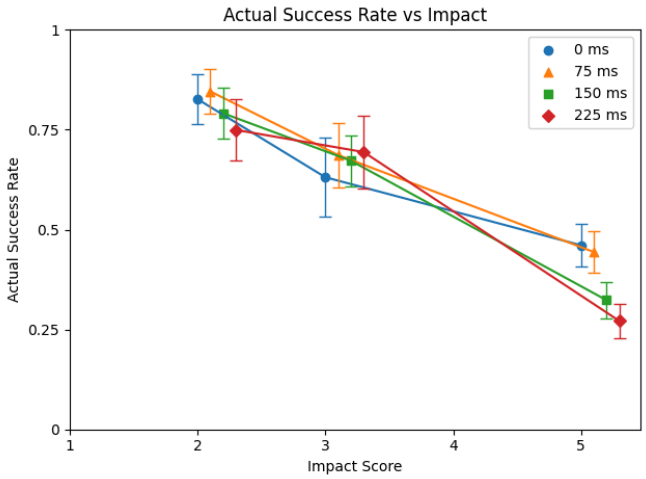


Figure 49: Actual Success Rate vs Impact Per-Frametime Spike Duration. The error bars show the 95% confidence interval.

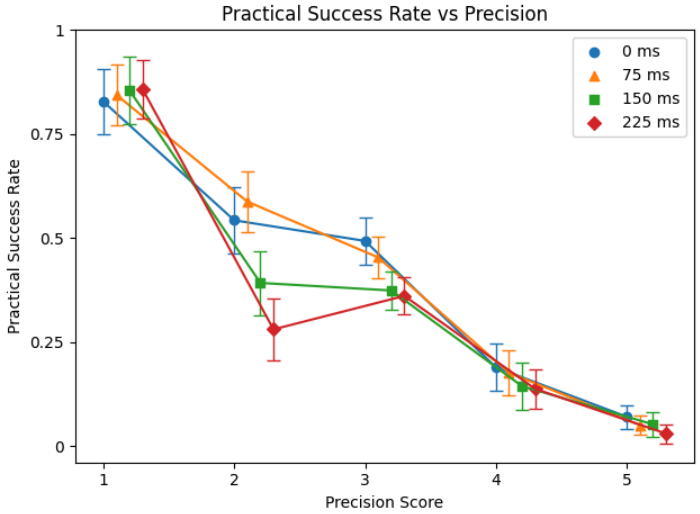


Figure 50: Practical Success Rate vs Precision Per-Frametime Spike Duration. The error bars show the 95% confidence interval.

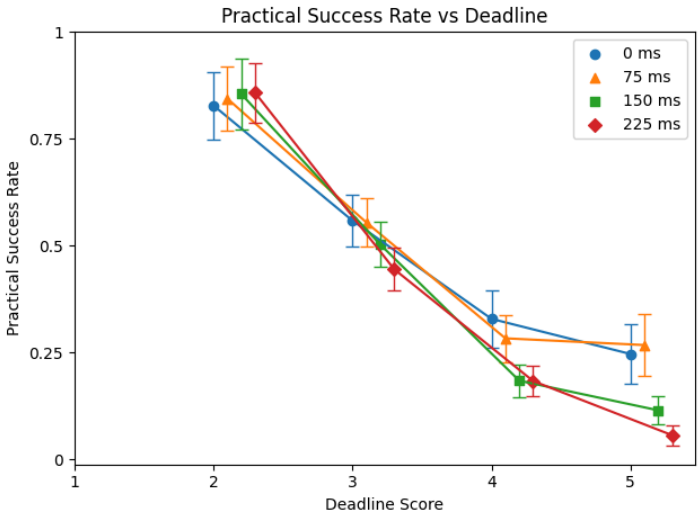


Figure 51: Actual Success Rate vs Deadline Per-Frametime Spike Duration. The error bars show the 95% confidence interval.

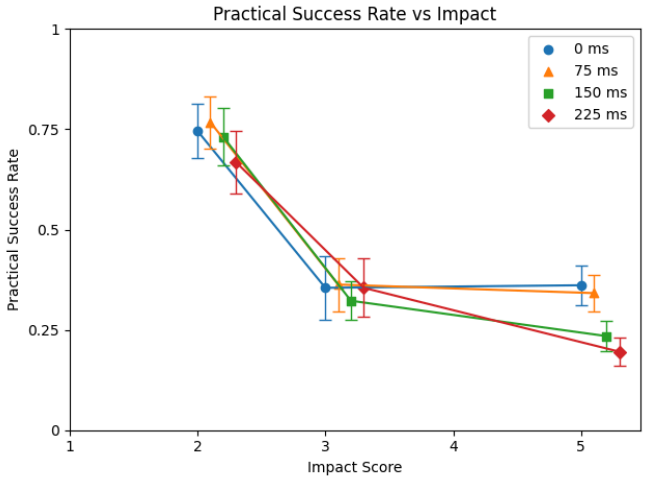


Figure 52: Actual Success Rate vs Impact Per-Frametime Spike Duration. The error bars show the 95% confidence interval.

5.3.4 Average Precision/Deadline/Impact Score as a Predictor of Success Rate and QoE

Figure 53, Figure 54, and Figure 55 show the average precision/deadline/impact score of a task compared to its success rate and QoE. There is a clear negative correlation between averaged score and actual and practical success rates. This suggests that averaging the precision, deadline, and impact scores of a task is a good indicator of its difficulty.

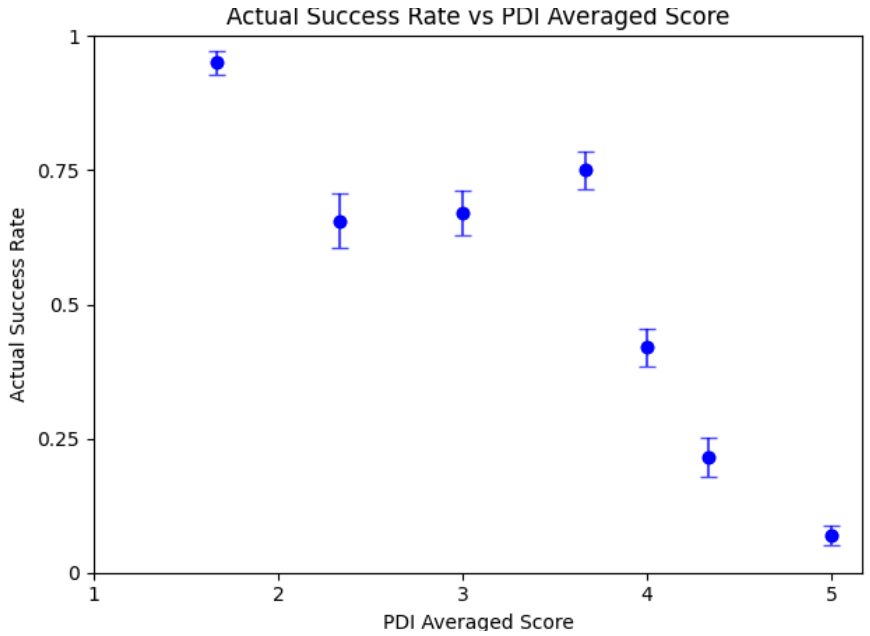


Figure 53: Actual Success Rate vs. Averaged Precision/Deadline/Impact Score.

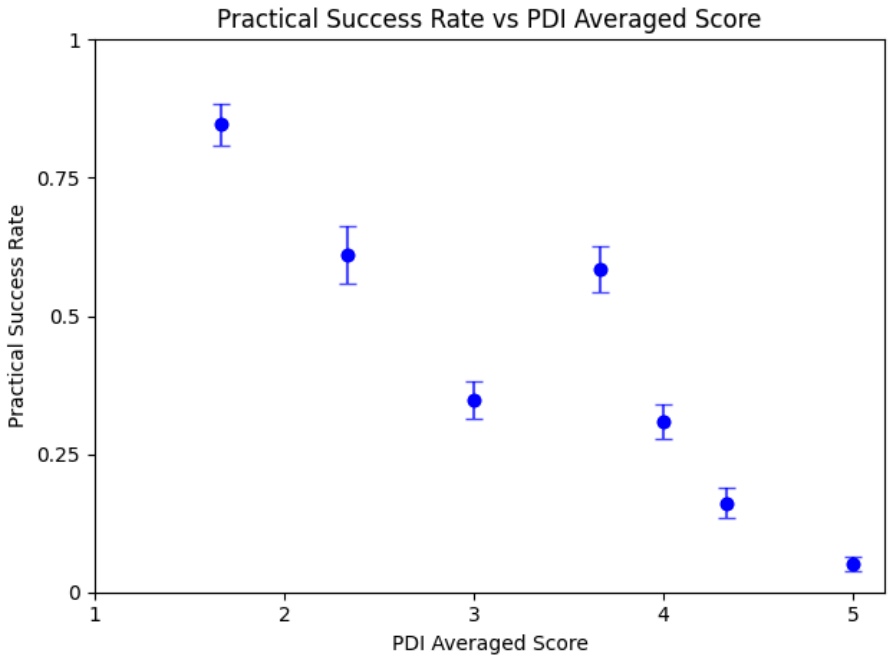


Figure 54: Practical Success Rate vs. Averaged Precision/Deadline/Impact Score.

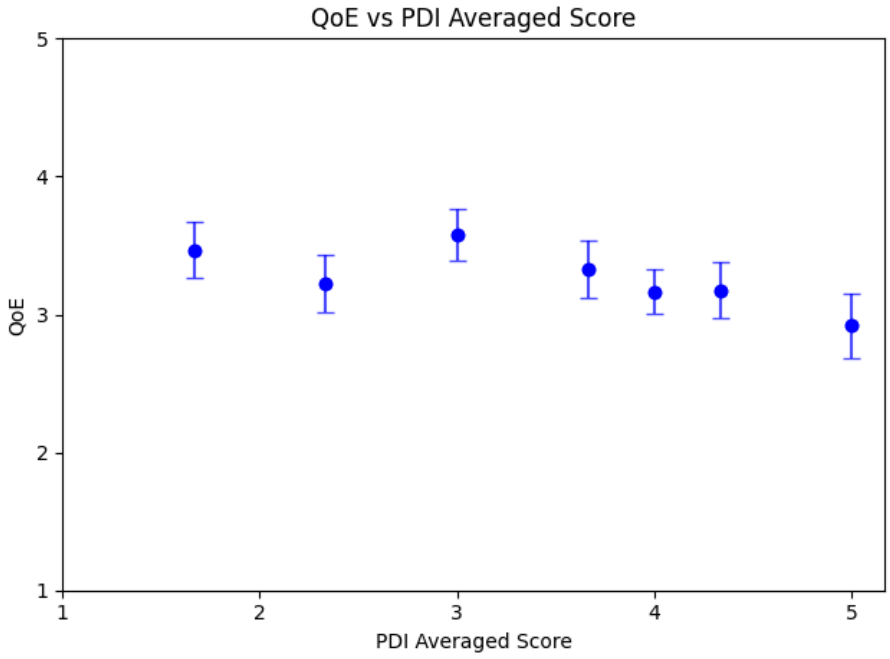


Figure 55: Average QoE Score vs. Averaged Precision/Deadline/Impact Score.

5.4 Summary

Through analyzing our data we can deduce that as frametime spike size increases, QoE scores decrease over all task groupings. However, the data show that the relationship between participant performance and frametime spike size is dependent on the individual task grouping. For the enemy and collectible groupings, performance did not have a significant change when frametime spike time increased, whereas, with jumping-related tasks there was a clear negative correlation. When revisiting our precision/deadline/impact rubric, it was apparent that it was a significantly better predictor of a task's sensitivity to frametime spike size as it relates to performance rather than for QoE.

6. Conclusion

This study explored the impact of frametime spikes on player performance and QoE during navigation-based tasks in 2D Platformers. We used an open source 2D platformer, SuperTux Classic, to isolate navigation mechanics as much as possible from other mechanics. The source code of the game was modified to add custom levels containing tasks and special hitboxes that induce lag to specifically research the effects of lag on navigation-based tasks.

We conducted a user study where 31 participants played 32 rounds of our game. Analysis of the data shows a correlation between frametime spike size and QoE, as the size of the spike increases, the QoE reported by the player decreases. The magnitude of this decrease is similar for every level type. From this study, it can be deduced that frametime spike size and QoE have a negative correlation. Similarly, the data shows a correlation between frametime spike size and

player performance, as the size of the spike increases, the performance of the player decreases. The magnitude of this decrease is most pronounced in jumping-based tasks. Item collection and enemy squishing tasks did not show a clear correlation. From this study it can be deduced that frametime spike size and player performance have a negative correlation only in jumping-based tasks. We also found that the precision/deadline/impact score was a promising predictor of a task's sensitivity to frametime spikes, whereas for QoE, it was not definitively effective.

7. Future Works

Our study has provided insight into relationships between frame stutters and their impact on player performance and quality of experience in 2D platformers. However, future researchers can take steps to further understand the effects of frame stutters on game players.

7.1 Limitations

Our project had limitations that can be improved upon in future research to potentially gain more insight into the relationships and correlations associated with our topic.

This study had 31 participants but correlations and relationships may be stronger and more definitive if more participants are tested in future adaptations of this study. Having more users broadens the demographics of the participant pool.

This study also is centered around SuperTux Classic, a 2D platformer, which is one particular game genre. Based on the demographics for our study participants (Appendices H, I, J, K, & L), 71% of players were familiar with 2D platformers before playing, this could potentially impact their performance in the game, not necessarily as a consequence of the frame stutter that was implemented.

Outreach to prospective participants could be improved in a future rendition of this study. Our team opted for batch emails to relevant user groups and messages into private forums within the clubs and organizations that we are a part of. While this method did work, it can be expanded to include posters and digital advertisements as well. This could draw more participants from

more diverse subsets of the community rather than those likely within the major organizations that the members of our team are involved in.

7.2 Challenges

The game build our team made for the playtesting study had issues with incrementing the player ID which we caught after the second participant. This issue caused us to run the other 29 through the game engine instead of a standalone build. This could have performance implications on the game. However, our team ran performance analyses across both the standalone build and the game engine and found no significant performance difference between them (Appendices M & N). However, in a future implementation of this study a functional standalone build for the user study could help eliminate potential performance impacts by the game engine.

There was an issue with the way our logging system was constructing the summary log during the playtesting study. For the first five participants, our team had to manually reconstruct the summary log, which is directly derived from the three other logs: frame log, event log, and QoE log. Though this challenge had no negative implications for our data, manually reconstructing the summary logs for these participants slowed down our data analysis process. Ideally, in a future project, the logging apparatuses would be tested more thoroughly and may avoid the unnecessary time spent to rectify the issue.

7.3 Additional Research

This research can be further expanded to gain a new understanding of the ways that frame stutter impacts performance and quality of experience in video games. These have been broken down into short-term, medium-term, and long-term additions.

7.3.1 Short-Term

Further short-term research (a month or less) includes performing the playtesting study again with more participants. This would require the researchers to replicate the previously outlined methodologies and perform the study again. The results could be compared between each study. It would allow the researchers to see if these results further support or challenge the conclusions drawn from our study. This can be done by using the demographics survey (Appendix G), reaction time test (Appendix C), downloading the build (Appendices O & P) used for this study, and playing on a similar machine as described in the methodologies section.

Also, future researchers could further analyze our data to connect the demographics survey and reaction time data further with the performance and quality of experience data. This may uncover new potential relationships and correlations associated with a participant's previous gaming experience. This could open up new opportunities for research into the effects of frame stutters. This could be done by referencing the raw data from this study and performing further data analysis which can be done in the short term without having to rerun the playtesting study.

7.3.2 Medium-Term

Further research in the medium-term (a few months) involves modifying the game further to include a different implementation of the lag spike. Network lag could be introduced by running the game through a client-server connection. The results could be compared to the results of this study to investigate potential similarities and differences between the performance and quality of experience of the network latency versus the frame stutter introduced in this study.

Also, a future research team could adapt the methodologies within this study to a different 2D platformer which includes more complex navigation components, such as wall jumping, flying, or sliding. This could provide further insight into the implications of frame stutters on a wider breadth of navigation-based tasks in 2D platformers. This can be done by either adapting the SuperTux Classic build that was used for this study or perhaps finding a different 2D platformer with more robust navigation components.

Finally, research can be done to create additional custom levels in SuperTux Classic to investigate other tasks that were not included in our study. For example, studying how frame stutters impact a player's ability to squish more complex enemies. We also did not explore having players collect coins. This could be used to encourage participants to jump in a particular trajectory outlined by coins. These tasks could also provide more data on the precision/deadline/impact rubric. More research can be done into how each of the scores contributes to projecting player performance and quality of experience as it relates to frametime spikes. One thought could be that it is the highest number of the three that trumps the relationship; but more data would help better clarify those relationships.

7.3.3 Long-Term

Further research in the long-term (over six months or more) includes testing 3D platformers, which feature complex navigation components. These results could be compared to the results found in this study to see how the results differ between each genre. This can be accomplished by creating or modifying an existing 3D platformer similar to this study.

Further research can also be done by expanding this study's methodologies to other games in other genres such as first-person shooter, role-playing game, massively multiplayer online games, etc. This may provide results that can be compared to those found in this study to uncover potential insights into the way different game genres are impacted by frame stutters. Also, it could be interesting to see how the QoE scores differ between different game genres when faced with the same frame stutter durations used in our study (0ms, 75ms, 150ms, and 225ms).

References

Alzter. 2023. *SuperTux Classic*. Game [PC]. (21 March 2023). Altzer.

CD Projekt Red. 2020. *Cyberpunk 2077*. Game [PC]. (10 December 2020). CD Projekt.

Claypool, K., & Claypool, M. (3 April 2007). *On frame rate and player performance in first person shooter games*. Multimedia systems.

Claypool, M., & Claypool, K. (1 November 2006). *Latency and player actions in online games*. Communications of the ACM.

Claypool, M., & Claypool, K. (26 April 2009). *Perspectives, frame rates and resolutions: it's all in the game*. In Proceedings of the 4th International Conference on Foundations of Digital Games - Orlando, USA.

Godot. Version 3.5.2. Godot Foundation. (7 March 2023).

Godot's architecture diagram. Godot Engine documentation. (2024). https://docs.godotengine.org/en/4.3/contributing/development/core_and_modules/godot_architecture_diagram.html

Liu, S., & Claypool, M. (28 April 2022). *The impact of latency on navigation in a first-person perspective game.* In Proceedings of the CHI Conference on Human Factors in Computing Systems - New Orleans, USA.

Liu, S., Kuwahara, A., Scovell, J. J., & Claypool, M. (April 2023). *The effects of frame rate variation on game player quality of experience.* In Proceedings of the CHI Conference on Human Factors in Computing Systems - Hamburg, Germany.

Xu, X., & Claypool, M. (17 April 2024). *User study-based models of game player quality of experience with frame display time variation*. In Proceedings of the 15th ACM Multimedia Systems Conference - Bari, Italy.

Appendices

WORCESTER POLYTECHNIC INSTITUTE

100 INSTITUTE ROAD, WORCESTER MA 01609 USA

Institutional Review Board
FWA #00030698 - HHS #00007374

Notification of IRB Approval

Date:	17-Oct-2024
PI:	Mark L Claypool
Protocol Number:	IRB-25-0248
Protocol Title:	First-Person Science: Exploring the Impact of Frame Stutter on Navigation-Based Tasks in 2D Platformers
Approved Study Personnel:	Claypool, Mark L~Chartier, Connor T~Anderson, Nathan A~Asher, Jonathan D~
Effective Date:	17-Oct-2024
Exemption Category:	3
Sponsor*:	

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

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

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

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

*if blank, the IRB has not reviewed any funding proposal for this protocol

Appendix A. IRB Approval For User Study.

WORCESTER POLYTECHNIC INSTITUTE

100 INSTITUTE ROAD, WORCESTER MA 01609 USA

Institutional Review Board
FWA #00030698 - HHS #00007374

Notification of IRB Approval

Date: 27-Jan-2025

PI: Mark L Claypool
Protocol Number: IRB-25-0248
Protocol Title: First-Person Science: Exploring the Impact of Frame Stutter on Navigation-Based Tasks in 2D Platformers

Sponsor*:

The WPI Institutional Review Board (IRB) approves the modification submitted on to the above-referenced protocol.

This modification does not extend the expiration date of your approval. The previous approval remains in effect from 17-Oct-2024 until 16-Oct-2025 unless terminated sooner (in writing) by yourself or the WPI IRB. If the research is to continue past 16-Oct-2025, you must submit a Study Renewal form to the IRB via InfoEd, at least 30 days prior to expiration.

Modification:

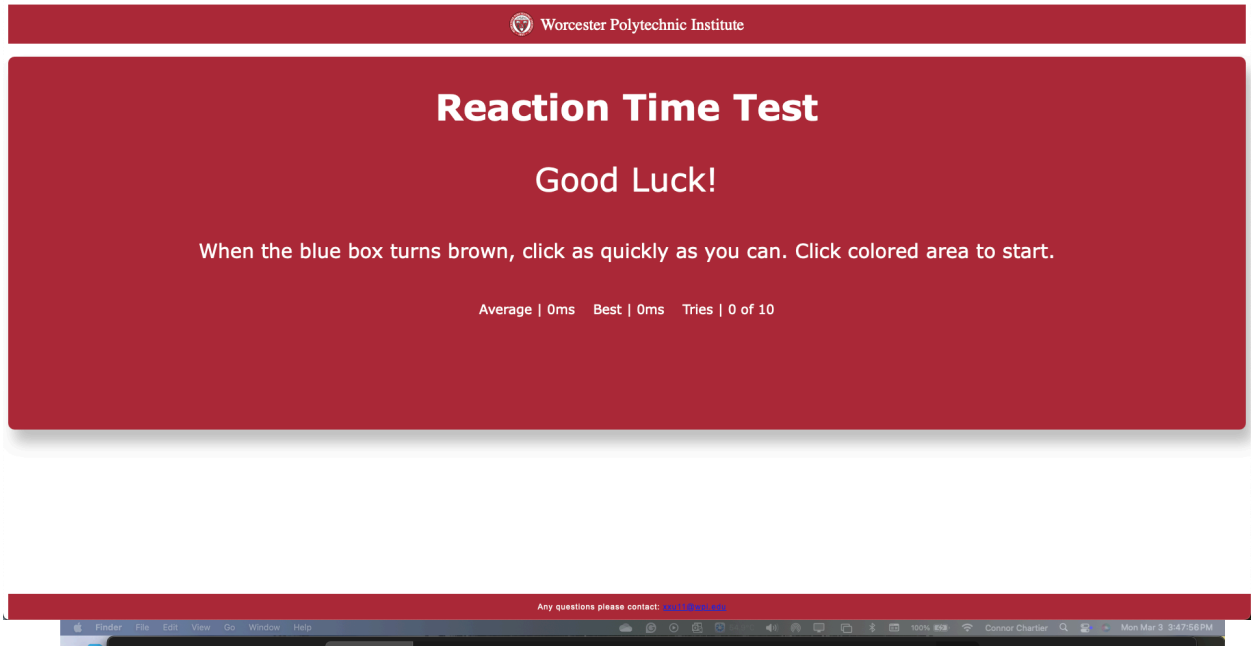
1) Our first amendment includes a brief reaction time test that will occur before the user plays our game. This consists of 10 rounds that evaluate participant reaction time in milliseconds. This creates a baseline for the reaction skills of our participants. This information can allow us to analyze how reaction time may be related to in-game performance.

2) Our second amendment includes a clause in compensation for participants about offering IMGD playtesting credit for IMGD courses. This will allow participants to use the follow-up email as proof of IMGD playtesting and consequently receive credit.

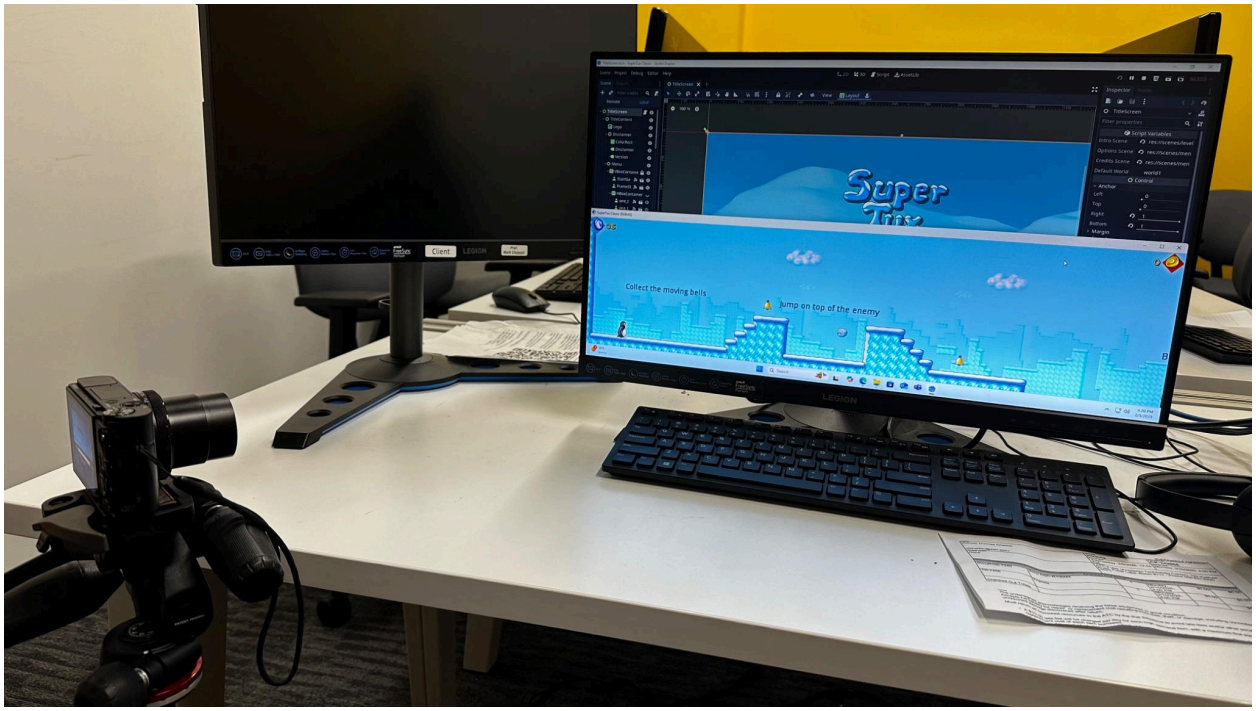
Please contact the IRB at irb@wpi.edu if you have any questions.

*if blank, the IRB has not reviewed any funding proposal for this protocol

Appendix B. IRB Approval For User Study Addendum. This is for our study to be approved to administer a reaction time test and provide IMGD playtesting credit for participants.



Appendix C. Reaction Time Test. Participants would click the colored area when the box changed colors. This would be repeated 10 times.



Appendix D. Recording Setup For 1000 FPS Videos.

Here is a consent form to review and fill out. You are able to stop playing at any time, for any reason and still receive full credit. (give consent form)

Here is a demographics survey to fill out. All of the questions are optional, so if you're not comfortable filling out a question, you can skip it. (give demographics survey)

Now you will be given a reaction time test to complete. The instructions will be provided on screen. (conduct the reaction time test)

You'll be playing SuperTux Classic, which is a 2D platformer like Super Mario. You will be playing a total of 34 rounds with varying levels of smoothness. The first two will be practice rounds. The first practice round will be the smoothest possible experience, and the second practice round will be the choppiest possible experience. This game has sound and if you wouldn't mind, we have headphones here for you to listen with.

You can use WASD or the arrow keys for movement and spacebar to jump. After each round, you'll be asked to rate the smoothness of the round.

We don't care how well you perform. If you see us watching you during the playtest, we're just looking for bugs.

(start the study)

Appendix E. Playtesting Script.

Informed Consent Agreement for Participation in a Research Study

Investigators: Jonathan Asher, Nathan Anderson, Connor Chartier, Professor Mark Claypool
Contact Information: gr-first-person-science-mqp@wpi.edu, claypool@wpi.edu
Title of Research Study: First Person Science: Exploring the Impact of Frame Stutter on Navigation-Based Tasks in 2D Platformers
Sponsor: NVIDIA

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:
This study measures the effect of frame stutters on player performance and quality of experience in 2D platformers in order to create a generic rubric to measure the frame stutter sensitivity of different tasks in games. These results could help future game developers more effectively implement lag compensation methods within games with significant navigation components.

Procedures to be followed:

1. The participant will be assigned a participant ID number when they arrive at the playtesting room
2. The participant will be asked to be seated an make themselves comfortable in front of a computer with a keyboard and mouse
3. The participant will be prompted to review the consent form
    a. If they accept and sign, we will continue the procedures
    b. If they decide not to sign the consent form, they are free to leave
4. The participant will be asked to take a 2-3 minute demographics survey regarding preview gaming experiences and enter their participant ID number
5. The participant will be asked to complete a reaction time test
6. The participant will be provided a sheet with the basic controls for the game
7. The participant will be prompted to complete 2 practice rounds of the game
8. The participant will play 20 main playtesting rounds
    a. Completing a one-question post round survey after each round

APPROVED BY
WPI IRB-1
1/27/2025

9. The participant will gather their belongings and exit the playtesting room

Risks to study participants:
There is no risk to study participants beyond that of their everyday activities as a student.

Benefits to research participants and others:
This research will potentially allow for game developers to better understand where frustration points are maximized from lag, and may be able to better optimize their games in key areas. This would ideally improve a gamer's experience.

Record keeping and confidentiality:
We will not be associating participant data with their name. We will be assigning each participant an ID number and all survey responses will be associated with the participant ID number. When participants sign up, we will be asking for their WPI email in order to be able to contact them if they win the raffle prize ($25 Amazon gift card). 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 no risk to study participants that is beyond that of their everyday activities as a student. However, you do not give up any of your legal rights by signing this statement.

Cost/Payment: (3) $25 gift cards will be raffled to study participants, who have equal chances of winning. No participant can win more than one gift card. Participants will also be eligible to receive IMGD playtesting credit for IMGD courses.

For more information about this research or about the rights of research participants, or in case of research-related injury, contact:
Nathan Anderson: nanderson2@wpi.edu
Jonathan Asher: jdasher@wpi.edu
Connor Chartier: ctchartier@wpi.edu
Professor Mark Claypool: claypool@wpi.edu
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

APPROVED BY
WPI IRB-1
1/27/2025

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.

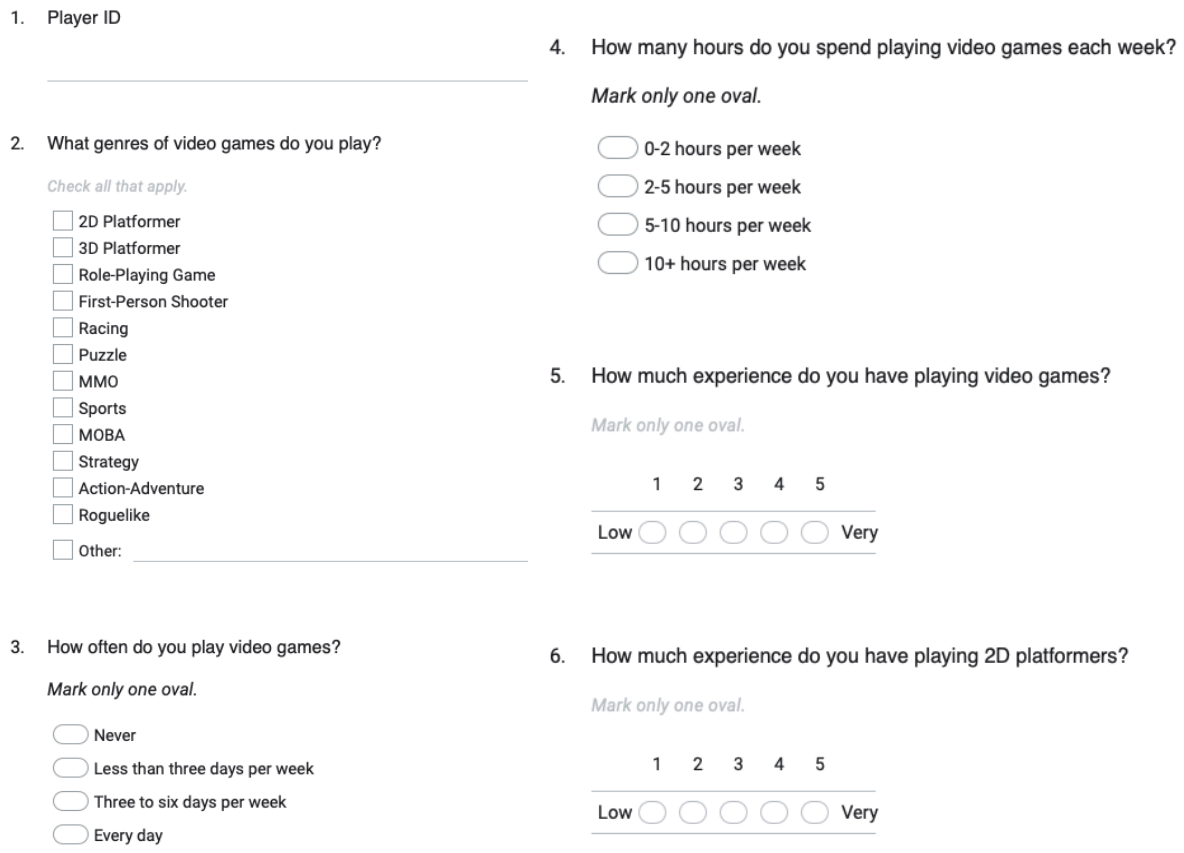
_____ Date: _____
Study Participant Signature

_____
Study Participant Name (Please print)

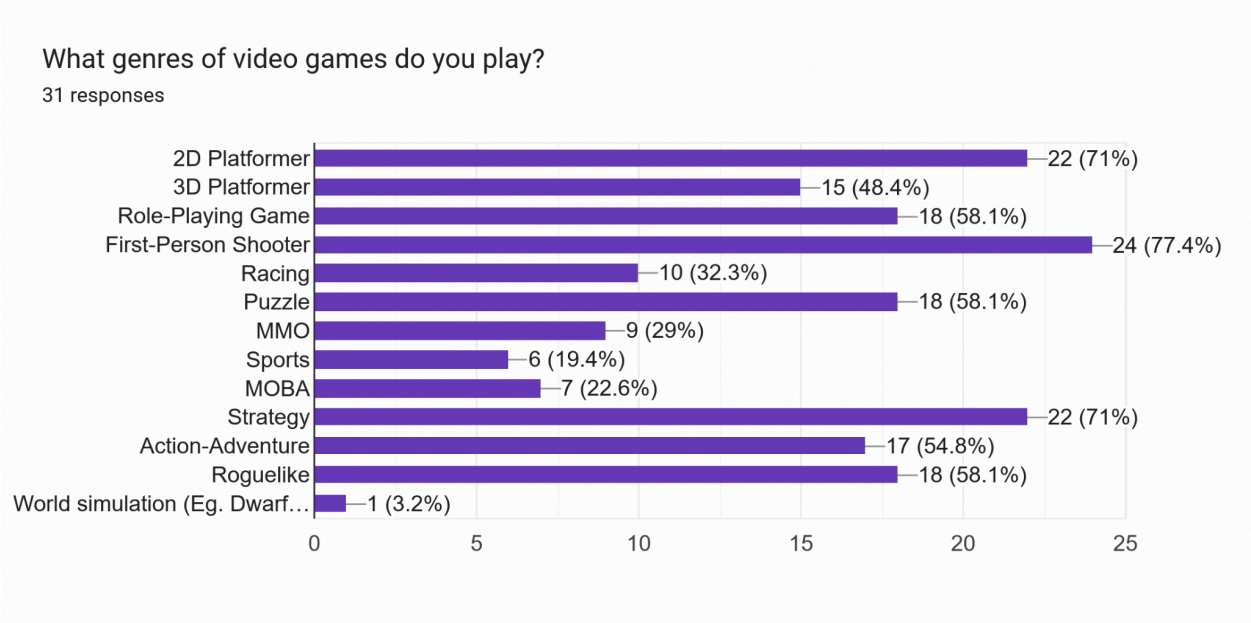
_____ Date: _____
Signature of Person who explained this study

APPROVED BY
WPI IRB-1
1/27/2025

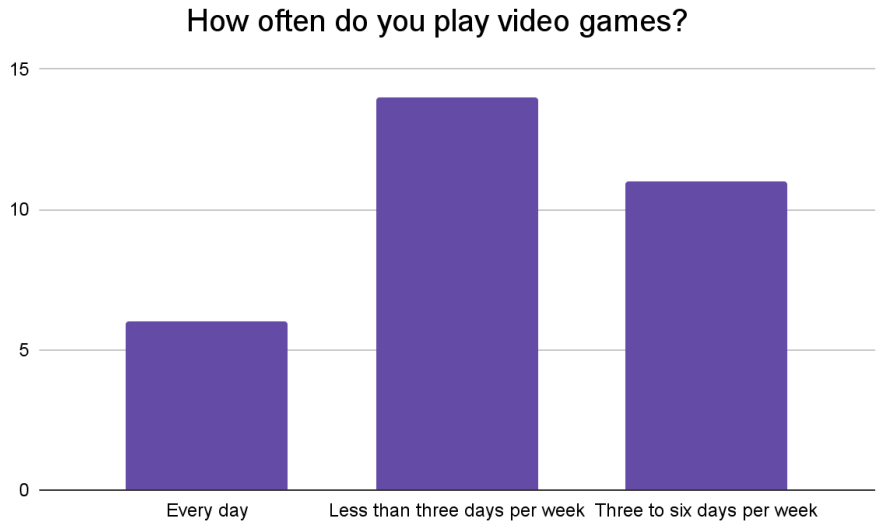
Appendix F. Informed Consent Form. Given to all participants.



Appendix G. Demographics Survey. Screenshots of the Demographics Survey feature six optional questions for participants to answer.



Appendix H. Demographics Question 1 Results.



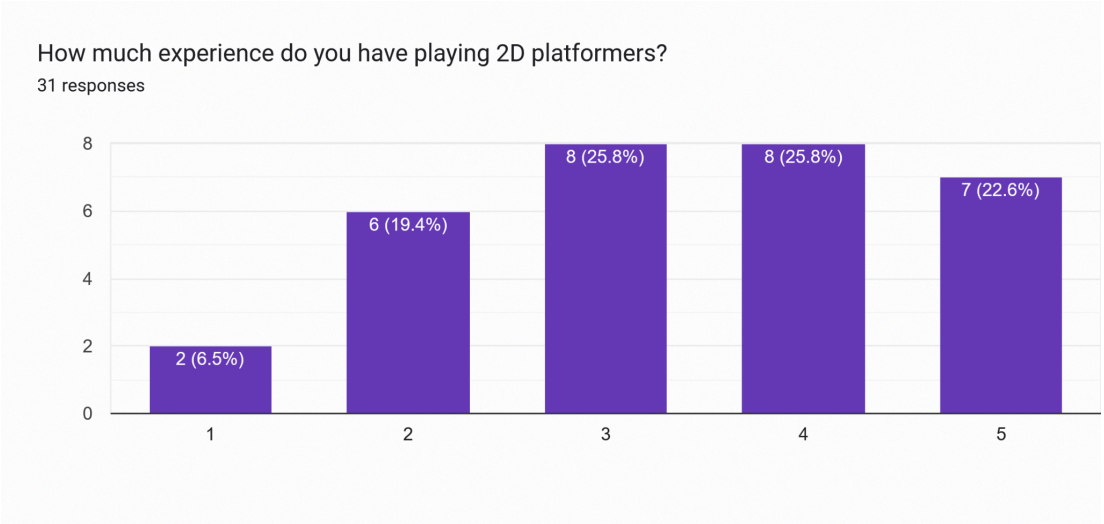
Appendix I. Demographics Question 2 Results.



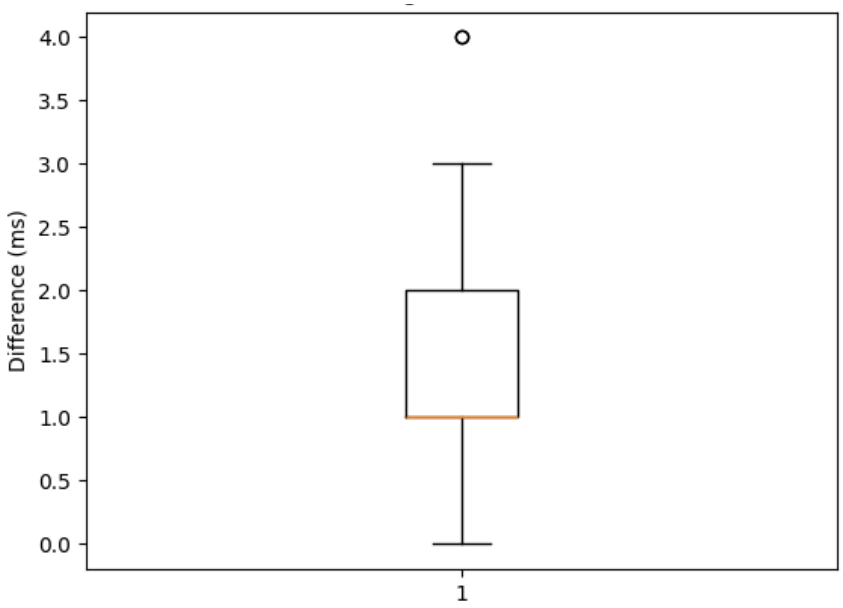
Appendix J. Demographics Question 3 Results.



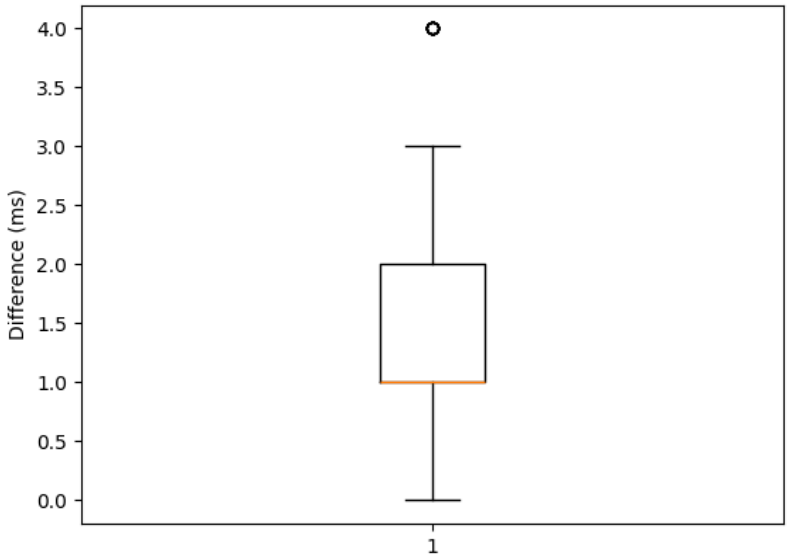
Appendix K. Demographics Question 4 Results.



Appendix L. Demographics Question 5 Results.



Appendix M. Distribution of Lag Differences for Executable. This displays the difference in expected vs actual lag duration measured in a standalone executable version of our modified game.



Appendix N. Distribution of Lag Differences for Game Engine. This displays the difference in expected vs actual lag duration measured in the game engine of our modified game.

https://github.com/nathan-anderson16/SuperTux-Classic

Appendix O. SuperTux Classic Github Link. Build instructions are within the repository readme file.

https://github.com/nathan-anderson16/SuperTux-Analysis

Appendix P. Data Analysis Github Link.