Impact of Frametime Spikes on Performance and Quality of Experience in Platformer Games

Samin Shahriar Tokey
Worcester Polytechnic Institute
Worcester, MA, USA
sstokey@wpi.edu

Ben Boudaoud
NVIDIA
Durham, North Carolina, USA
bboudaoud@nvidia.com

Josef Spjut
NVIDIA
Durham, North Carolina, USA
josef.spjut@gmail.com

Mark Claypool
Worcester Polytechnic Institute
Worcester, MA, USA
claypool@wpi.edu

Abstract

Frametime spikes can disrupt gameplay in games, affecting both player performance and experience, but the effects of these spikes on navigation based tasks is not well-studied. This work investigates how frametime spikes impact players performing navigation-focused tasks in a platformer game. An open-source platformer game, SuperTux Classic, was modified to deliberately create spikes in frametimes when players performed certain actions, while recording performance and assessing quality of experience (QoE). Thirty-one participants completed eight distinct navigation-based tasks, each with predetermined spike durations. Analysis of the data shows that the effects of frametime spikes on player performance depends on the task, but the effects on QoE are largely independent of task.

CCS Concepts

• **Human-centered computing** → *User studies*; Laboratory experiments; • **Applied computing** → **Computer games**.

Keywords

2D platformer, frametime variation, framerate, spikes, stutters, lag, QoE, quality of experience, visual perception

ACM Reference Format:
Samin Shahriar Tokey, Ben Boudaoud, Josef Spjut, and Mark Claypool. 2026. Impact of Frametime Spikes on Performance and Quality of Experience in Platformer Games. In *Foundations of Digital Games (FDG '26), August 10–13, 2026, Copenhagen, Denmark*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3815598.3815634

1 Introduction

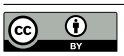
Advancements in computer hardware and supporting software continue to shape the gaming industry, allowing developers to create more visually impressive and immersive experiences [7]. However, these graphical improvements often come at the cost of performance, especially in modern games requiring high system specifications. Maintaining a consistent frame rate can be a challenge as the time to render individual frames can vary with the scene complexity and scene changes and can be exacerbated by in-game computations, with computation-based dips in frame rate that occur during a critical player's interaction being especially problematic. As graphics become more complex, maintaining consistent frame rates and minimizing stutters becomes increasingly challenging.

In cloud-based games, variation in network conditions can introduce visual stuttering similar to dips in local frame rates [11], disrupting the smoothness of gameplay and affecting the player's quality of experience (QoE). Understanding the impact of frame stutters on player performance and QoE can inform the design of more connection-quality resilient cloud gaming platforms.

While the effects of frame rate on player performance and QoE have been studied extensively, particularly in first-person shooter (FPS) games where precision and responsiveness are critical [3, 4, 8, 14, 16], the specific effects of frametime spikes on navigation-based tasks in games remain underexplored. Unlike FPS games, which emphasize aiming and shooting mechanics, 2D platformers are predominantly navigation-centric, focusing on moving through the environment from point A to point B while avoiding obstacles. This navigation is usually achieved through running and jumping. For example, Figure 1 from SuperTux Classic shows a task where the penguin (player) must jump across a gap to reach a bell. The dotted arrows illustrate the jumping path required to clear the gap.

Navigation tasks are fundamental in many game genres, yet their sensitivity to frame stutters has not been thoroughly investigated. Our study aims to bridge this gap by examining the effects of frametime spikes on 2D navigation-based tasks. We implemented a task difficulty rubric to categorize different navigation challenges and designed a user study with frametime spikes of varying durations (0, 75, 150, and 225 ms). The frametime spikes were introduced using a "random delay" method, where spikes occur a random amount of time after a trigger event in order to make them less predictable to the users.

Participants played two practice rounds to familiarize themselves with the best-case (0 ms) and worst-case (225 ms) frametime spike conditions, followed by 32 collection rounds. Each round lasted approximately 30 seconds and featured one of eight navigation tasks, each tested with all four frametime spike durations. The



FDG '26, Copenhagen, Denmark

ACM ISBN 979-8-4007-2495-4/26/08
https://doi.org/10.1145/3815598.3815634

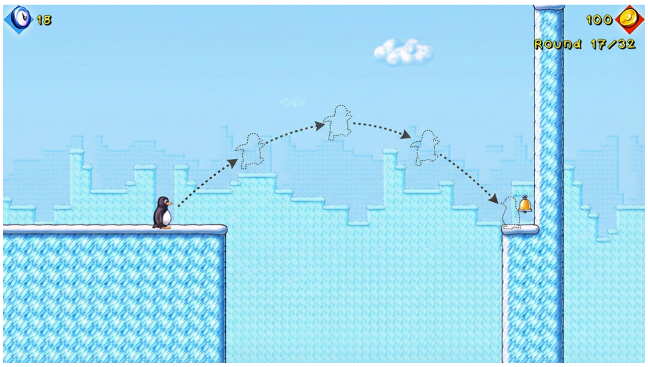


Figure 1: Super Tux Classic in-game screenshot. The player guides the penguin from the left platform across the gap to the right ledge. Touching the bell completes the jump.

objective was to complete as many tasks as possible within the time limit.

The rest of this paper is organized as follows: Section 2 summarizes related research; Section 3 outlines our methodology – assessing the effects of frametime spikes for various in game tasks through a user study; Section 4 analyzes the results from the user study; Section 5 discusses our key findings and their implications; Section 6 identifies some study limitations; and Section 7 summarizes our conclusions and suggests possible future work.

2 Related Work

This section describes related work in three areas: frame rates and games, frametime variation and games, and navigation-based tasks in games.

2.1 Impact of Frame Rate on Games

Researchers have explored how frame rate impacts player performance and experience. Claypool and Claypool [3] analyzed how different frame rates impact player performance across first-person, third-person, and isometric perspectives. Performance declined significantly below 15 f/s (66.6 ms frame time), particularly in tasks requiring quick, precise actions such as shooting. Users performed slightly better at 15–30 f/s (33.3–66.6 ms frame time), with noticeable gains in fast-response tasks compared to movement. Claypool et al. [4, 5] found that frame rate had a greater impact on player performance and perceived game quality than resolution. In contrast, for video playback, resolution was more critical.

We [16] evaluated effects of frame rate for up to 500 f/s (2.0 ms frame time) in a first-person shooter game. We found that player performance did not improve significantly above 60 f/s, even though players noticed and enjoyed frame rate all the way up to 500 f/s. We [18] further showed that in FPS games, frametime spikes during aiming reduce perceived smoothness and player performance, while graphical fidelity mainly affects perceived visual quality, with spikes lowering both perceived quality and smoothness.

Janzen and Teather [8] showed that performance at 60 f/s (16.6 ms frame time) was about 14% higher than at 30 f/s (33.3 ms frame time), while 45 f/s (22.2 ms) and 60 f/s did not differ significantly. They also found that frame rate had a stronger influence than latency alone would suggest.

Spjut et al. [14] studied how latency and refresh rates affect FPS game performance, focusing on esports athletes. They found that lower latency significantly improved both single-click and tracking tasks. Higher refresh rates (120–360 f/s) also boosted performance, especially in tracking tasks that require continuous aiming. The main advantage of higher refresh rates was reducing latency, highlighting its importance in competitive gaming.

Wang et al. [19] showed that frame rate in virtual reality significantly affected both user experience and task performance in a "Fruit Cutting" game, with associated impacts and benefits to motion sickness across 60-180 f/s.

QoE models have incorporated frame rates to predict gaming quality. Zadtootaghaj et al. [21] found that maintaining 25 f/s (40 ms frame time) at lower bitrates minimized visual issues, while lower frame rates caused severe jerkiness and fatigue.

2.2 Frametime Variation and Games

Several studies have examined the impact of frametime variation on QoE in cloud gaming. Rossi et al. [13] found that mobile cloud game streams react differently to packet loss, round-trip time, and delay jitter — all of which affect frametime variation — compared to traditional online games. Suznjevic et al. [15] reported that added variation in GeForce Now can force the system to reduce frame rate and resolution to their lowest levels.

QoE models have also incorporated frametime variation to predict gaming quality. Xu et al. [20] confirmed that while average frame rate often correlates with QoE, frametime variability and frametime spike severity are stronger predictors of player experience, explaining about 90% of the variation in gaming quality. Klein et al. [9] demonstrate that even minor frametime variability (on the order of a few milliseconds) can degrade perceived smoothness in gameplay, without producing a measurable drop in player performance when comparing equally smooth conditions. Similarly, network jitter was shown by Quax et al. [12] to have minimal impact on player QoE in an FPS until latency grows large (beyond roughly 60 ms), at which point both performance and enjoyment deteriorate noticeably.

Event-specific frametime spikes have also been shown to reduce player performance and experience in an FPS game when they occur during high-impact events such as targeting [17].

2.3 Navigation-Based Game Tasks

Liu and Claypool [10] examined the effects of local and network latency on player performance and QoE in a navigation-focused hide-and-seek game. They found that while both types of latency degraded performance and QoE as delay increased, there was no significant difference between them. These findings motivated our investigation into how frametime spikes affect navigation-based tasks.

Our study examines the impact of large frametime variations (stutters or spikes) in a 2D platformer game. Unlike previous research that considered random stutters, we focus on their effects during specific in-game navigation tasks.

3 Methodology

To investigate the effects of event-based frametime stutters in a 2D platformer game, we conducted a user study with a modified version of the open source platformer game called SuperTux Classic.[1]

3.1 Game Overview

SuperTux Classic is a 2D platformer game where players control Tux, a penguin avatar, to navigate through different levels full of smaller tasks (e.g., jumping a large gap, collecting a coin). SuperTux has also been used in other academic studies [1, 2, 6]. In the game, players move Tux left or right and jump to overcome obstacles. For our study, we designed eight types of levels using the built in level designer, each level focusing on a single task. They are:

- *Collecting Item: Moving.* The player must pick up an item that moves steadily before it disappears (see Figure 2a).
- *Collecting Item: Bouncing.* The player must collect an item that bounces before it disappears (see Figure 2b).
- *Squish Bouncing Enemy.* The player must jump on a bouncing enemy to defeat it (see Figure 2c).
- *Large Gap, Large Platform.* The player must jump over a large gap and land on a large, stable platform (see Figure 3c).
- *Small Gap, Medium Platform.* The player must jump over a small gap and land on a medium-sized platform (see Figure 3a).
- *Medium Gap, Small Platform.* The player must jump over a medium gap and land precisely on a small platform (see Figure 3b).
- *7-Jump.* The player must execute a multi-step sequence of directional jumps in a 7-shaped pattern (see Figure 3d).
- *Z-Jump.* The player must execute a series of directional jumps forming a Z-shaped trajectory (see Figure 3e).

We designed custom levels to isolate specific tasks and control difficulty. For our study, participants play variations of these rounds with different settings with a goal of finishing as many tasks as possible before the round timer reaches zero. When a task is successfully completed, the penguin avatar respawns at the original spawn location for that round.

The game user interface (UI) shows the score, which increases by 100 every time the player successfully completes a round's objective. There are no direct score deductions for failure; instead, a failed attempt results in a time penalty due to the death animation, which delays the next attempt. The death animation plays for approximately 2 seconds. The UI also shows which round the player is currently in, alongside the round timer.

The game includes sound effects and background music; however, the music was disabled for this study. Headphones were provided to ensure users could hear the interaction sounds

Figure 1 shows a screenshot of *SuperTux Classic*. In this screenshot player needs to jump over the gap and collect the bell to finish the level. The screenshot also demonstrates all the UI elements present during gameplay.

Figures 2 and 3 illustrate the eight level designs used in our study. In all levels, the player begins at the flag icon and must reach the bell to complete the task. The red blocks are triggers for spikes (see Section 3.2), red blocks marked with an "X" represent lethal lava that causes the player avatar to respawn at the start. The first set (Figure 2) contains the three *interaction tasks*, where the player must collect a moving item (Figure 2a), collect a bouncing item (Figure 2b), or squish a bouncing enemy (Figure 2c). The second set (Figure 3) shows the *jump-based tasks*, which include basic single-gap jumps (Figures 3a–3c) and two multi-step platforming challenges: the 7-Jump (Figure 3d) and Z-Jump (Figure 3e).

[1]SuperTux Classic: https://github.com/Alzter/SuperTux-Classic

3.2 Modifications for Experiments

For our study, *SuperTux Classic* was instrumented to introduce controlled frametime spikes during several game events. Frametime spikes are thrown using a *random delay* design. Each task has a defined area of interconnected trigger cells, invisible to the user. The red blocks in Figure 3 show the trigger areas for each level. Entering one of these areas initializes a timer to a random (in range) value which starts counting down once the player leaves the cell block. The timer stops counting if the player stops moving and is reset if the player re-enters a trigger cell in order to prevent players from potentially "dodging" the effects of frametime spikes. Once the timer hits zero, a spike is thrown. This method of throwing frametime spikes is customizable in that spike location areas can be explicitly defined as well as spike duration, making it useful for a wide-variety of navigation-based tasks. To ensure the spike occurred before the task ended, the maximum timer value was less than the minimum completion time measured in pilot studies.

The frametime spike magnitudes were 0 ms, 75 ms, 150 ms and 225 ms, selected based on pilot tests. Moreover, based on preliminary measurements across a range of game systems, this range (0 to 225 ms) covers a higher spread of frametime spikes than occurs most often, where spikes are clustered more heavily on the lower end. The frame rate was set to 60 f/s for all rounds.

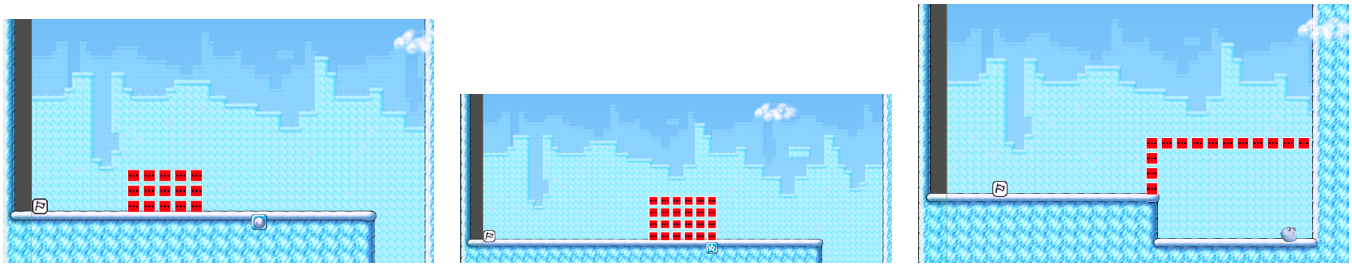
3.3 User Study

Each session began with two practice rounds – one with no frametime spikes to represent the smoothest condition and one with the maximum sized frametime spikes to represent the choppiest condition. Following the practice rounds, participants completed 32 main rounds. These rounds centered around eight level designs (Collect Moving Item, Collect Bouncing Item, Squish Bouncing Enemy, Large Gap–Large Platform, Small Gap–Medium Platform, Medium Gap–Small Platform, 7-Jump, and Z-Jump), each played under four different frametime spike settings. The level timer was set to 15 seconds for the Collect Moving Item, Collect Bouncing Item, and Squish Bouncing Enemy tasks, and to 30 seconds for the remaining tasks. The order of rounds was balanced among participants using a Latin square. Table 1 summarizes our experimental configuration.

A user ID was selected the appropriate round order from our Latin Square design. After each session, the user ID was incremented and saved to a file. For subsequent sessions, the stored ID was read from the file to determine the sequence of rounds.

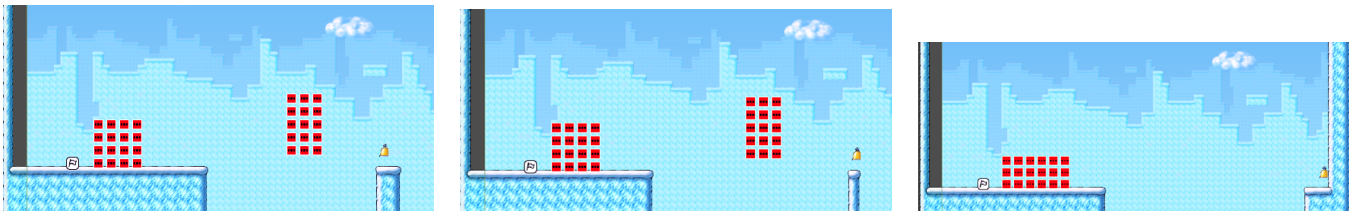
After the round ended, players were asked about their experience for that round via two questions:

- *Rate the Visual Smoothness of the Round*
  (Very Choppy) 1 to 5 (Very Smooth) via a slider.

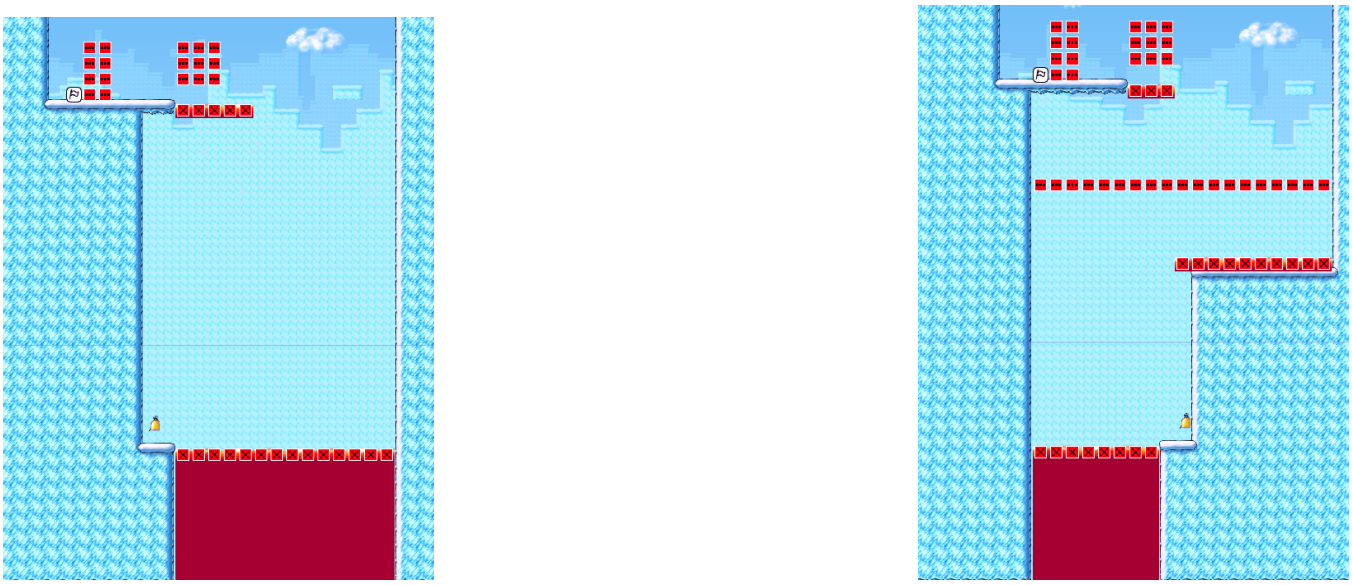


(a) Collecting Item: Moving (b) Collecting Item: Bouncing (c) Squish Bouncing Enemy

Figure 2: Interaction task level layouts. Red blocks represent stutter areas.



(a) Small Gap, Medium Platform (b) Medium Gap, Small Platform (c) Large Gap, Large Platform



(d) 7-Jump (complex pattern) (e) Z-Jump (complex pattern)

Figure 3: Top row: basic jumps. Bottom row: complex jumps. Red blocks represent stutter areas; red tiles with an X indicate lava.

- *Was the Smoothness of the Round Acceptable?* Yes / No via a radio button.

The acceptability question was tied to frametime spikes as to whether frametime spikes make the gameplay unacceptable. The game generated logs to record player responses, actions, performance metrics, and task outcomes, which were used in our analysis.

3.4 Demographics

A total of 31 players participated in the study. Table 2 summarizes their demographics. Numeric values are means with standard deviations in parentheses. Participants played video games approximately 5 days per week (M = 5.4, SD = 1.2) and spent about 8 hours per week gaming (M = 7.9, SD = 3.9). They rated their general gaming experience and 2D platformer experience on a scale from 1 (low) to 5 (high). On average, participants had moderate to high general gaming experience (M = 4.2, SD = 0.8) and moderate platformer

Table 1: Experimental configuration.

Parameter	Value
Spike magnitudes	0, 75, 150, 225 ms
Frame rate	60 f/s
Practice rounds	2 (no spikes, max spikes)
Main rounds	32 (8 tasks × 4 spike levels)
Round Duration	15 s (interaction), 30 s (jump)

experience (M = 3.4, SD = 1.2). Reaction times were quick (M = 211.9 ms, SD = 47.6), typical of experienced gamers.

3.5 Hardware

Our users participated on a PC running Windows 11 with Intel Core i7 8700 CPU, NVIDIA GTX 1080 GPU and 64 GB of RAM, with a 240 Hz, 1920×1080 Lenovo Legion Y25-25 monitor.

The local input latency was measured using a 1000 f/s camera (Casio EX-ZR100). The time between the button press and the screen change was calculated by analyzing the recorded frames. This measurement method was done 10 times on our client, yielding an average of 26.2 ms, ranging from a low of 20 ms to a high of 32 ms with a standard deviation of 3.2 ms.

Table 3 shows expected lag/stutter vs. measured lag. The expected lag is the programmed delay used to induce a frametime spike. The measured lag is the observed duration of that spike. For each spike magnitude, five measurements were taken using the 1000 f/s camera and averaged. All measured values were within one 60 Hz frame (16.7 ms) of the expected lag.

4 Analysis

This section presents an analysis of data from 31 participants, beginning with player performance and followed by player experience.

4.1 Performance

To evaluate player performance, we employ a *success rate* metric. In each round, players earn +100 coins for every success (e.g., completing a jump, defeating an enemy). The maximum successes possible for each round was determined based on the number of successes a player could achieve if they completed the task without mistakes as fast as possible. We then calculate the success rate for a player by converting the ratio of actual successes to maximum possible successes into a percentage.

Figures 4 and 5 presents the player performance analysis based on frametime spike magnitude. The x-axes are the frametime spike magnitudes and the y-axes are the success rates. Each point is the mean value across all users for that condition, shown with a 95% confidence interval.

From Figure 4, small frametime spikes (e.g., 75 ms) do not significantly impact success rates, but larger frametime spikes (150 ms or more) do lower performance. A one-way ANOVA confirmed a significant effect of spike magnitude on success rates, $F(3, 976) = 6.73$, $p < 0.001$. From Figure 5, tasks requiring more precise timing (e.g., *Small Gap, Medium Platform*) are affected by frametime spikes more so than tasks that do not.

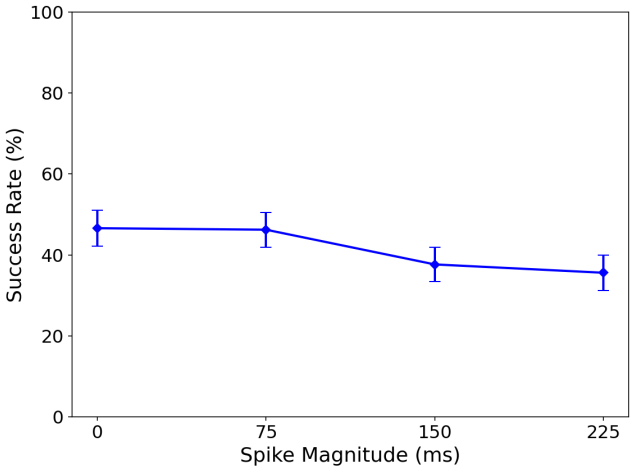


Figure 4: Overall success rate vs. spike magnitude. Mean with 95% confidence interval.

Table 4 shows the linear regression R^2 values for each task. The R^2 value for 'Collect Moving Collectable' is nearly 0, indicating that that frametime spikes have little impact on performance on that task. 'Collect Bouncing Collectable' and 'Squish Bouncing Enemy' have moderate R^2, indicating frametime spikes have some impact on performance for those tasks. For all the jumping tasks, R^2 values are high, indicating frametime spike sizes strongly explain the mean player performance for those tasks.

4.2 Visual Smoothness/Quality of Experience

After each round, participants rated the visual smoothness of the round from 1 (low) to 5 (high), which we refer to as Quality of Experience (QoE). Figures 6 and 7 display these results. The axes and data are as for Figures 4 and 5, but the data is the QoE rating instead.

Figure 6 shows that the average QoE is fairly high for frametime spikes from 0–75 ms, then steadily decreases as spike sizes get larger. A one-way ANOVA confirmed a strong effect of spike magnitude on QoE, $F(3, 1152) = 431.55$, $p < 0.001$. Figure 7 suggests tasks that require more precise timing experience a more dramatic drop in QoE once spike magnitudes surpass 150 ms.

Table 5 shows the R^2 values for task QoE versus spike magnitude. The high R^2 values show a strong relationship between frametime spike magnitude and QoE regardless of the task.

Figure 8 shows the relationship between users' acceptability ratings and their corresponding QoE scores. The x-axis is the QoE rating and the y-axis is the percent that users that found rounds with that QoE acceptable. Each point is the mean value averaged across all rounds in that QoE interval (in 0.5 bins), shown with a 95% confidence interval. Table 6 lists the numerical values for each bin (mean, confidence interval, and sample count). From the graph, rounds with a QoE above 4 were almost always acceptable, whereas those with a QoE below 2 were nearly always unacceptable, with a sharp change from unacceptable to acceptable in-between.

Table 2: Participant Demographics

User	Play Frequency (days/week)	Weekly Hours (hrs/week)	Gaming Exp. (1-5)	2D Platformer Exp. (1-5)	Reaction Time (ms)
31	5.38 (1.23)	7.86 (3.90)	4.23 (0.80)	3.39 (1.23)	211.86 (47.64)

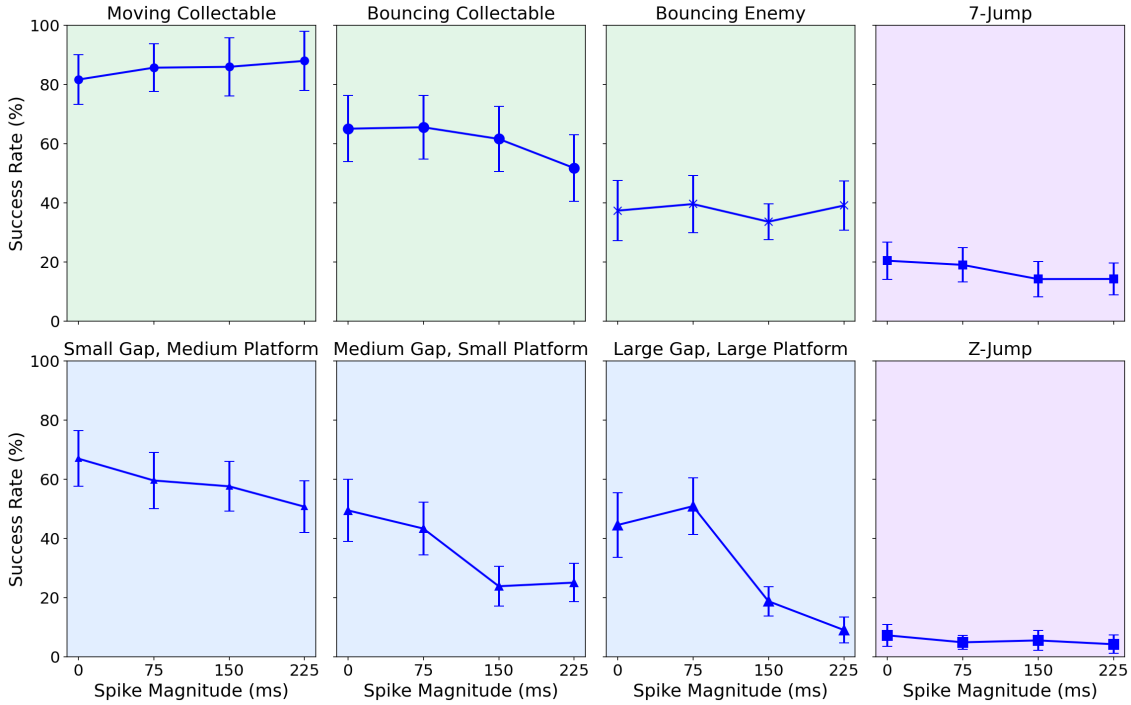


Figure 5: Success rate vs. spike magnitude for each task. Mean with 95% confidence interval. Shaded areas indicate task type: green for interaction, blue for basic jumps, and purple for complex jumps.

Table 3: Expected vs. measured frametime spike lag.

Expected (ms)	Measured (ms)	Difference (ms)	SD (ms)
75	91.53	+16.53	2.18
150	164.40	+14.40	1.72
225	234.39	+9.39	1.90

Table 4: R^2 values for player performance across tasks.

Task Type	R^2
Interaction Tasks	
Collect Moving Collectable	0.07
Collect Bouncing Collectable	**0.75**
Squish Bouncing Enemy	**0.64**
Basic Jump Tasks	
Small Gap, Medium Platform	**0.97**
Medium Gap, Small Platform	**0.91**
Large Gap, Large Platform	**0.78**
Complex Jump Tasks	
7-Jump	**0.91**
Z-Jump	**0.89**

These thresholds can be used in conjunction with data from Figure 6 to ascertain how frametime spikes must be limited by a game system in order to achieve acceptable QoE. For example, frametime spikes smaller than 50 ms will always be acceptable, whereas frametime spikes 225 ms or larger will always be unacceptable. This can be helpful for developers in achieving performance targets and for players in deciding upon system upgrades.

5 Discussion

Frametime spikes adversely affect both player performance and visual quality of experience (QoE), particularly when they happen during high-impact actions. In precisely-timed navigation-based tasks for 2D platformers, like jumping gaps, larger spikes lead to lower success rates. Whereas more loosely-timed tasks like collecting items are comparatively unaffected. This indicates that while frametime spikes typically degrade overall performance in

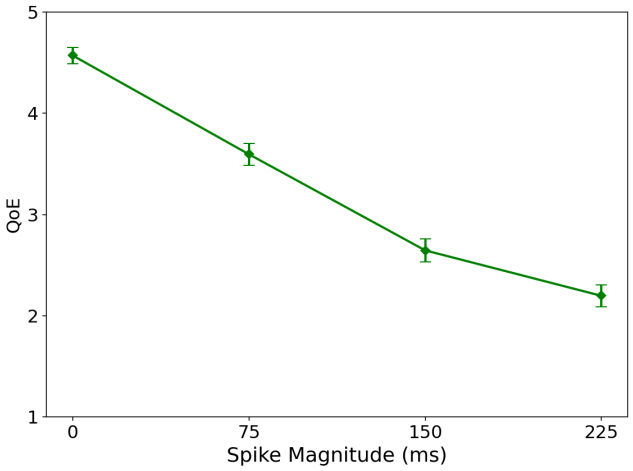


Figure 6: Overall QoE vs. spike magnitude. Mean with 95% confidence interval.

Table 5: Task and R^2 value for player experience.

Task	R^2
Interaction Tasks	
Collect Moving Collectable	**0.95**
Collect Bouncing Collectable	**0.98**
Squish Bouncing Enemy	**0.94**
Basic Jump Tasks	
Small Gap, Medium Platform	**0.93**
Medium Gap, Small Platform	**0.95**
Large Gap, Large Platform	**0.94**
Complex Jump Tasks	
7-Jump	**0.94**
Z-Jump	**0.99**

QoE Bin	Mean	95% CI	N
1.0	0.08	0.07	71
1.5	0.07	0.05	98
2.0	**0.13**	**0.06**	**135**
2.5	0.28	0.07	156
3.0	0.66	0.08	125
3.5	0.90	0.05	113
4.0	**1.00**	**0.00**	**123**
4.5	0.99	0.02	96
5.0	1.00	0.01	239

Table 6: Fraction acceptable per QoE bin. Bold values indicate the two key transition points highlighted in Figure 8.

navigation-based tasks, the magnitude of the effect is closely tied to the task's required timing precision.

In contrast, player perception of visual smoothness degrades regardless of the task's timing precision requirements. Even in tasks where player performance remains largely unaffected by frametime spikes, players still noticed a degradation in visual smoothness. Still, more precisely timed tasks, such as complex jumps with mid-air maneuvers, are somewhat *more* impacted by frametime spikes than less-precisely timed tasks in QoE. Since QoE scores above 4 were nearly always reported as acceptable, and 75 ms of delay had mean QoE scores just under 4, developers should aim to keep frametime spikes below about 50 ms to ensure their game(s) feel smooth to the player.

6 Limitations

Our study focuses on frametime spikes coupled to specific navigation tasks in a 2D platformer, like jumping and collecting items. However, other platformer actions, such as wall-jumping, dashing, gliding, and shooting, among others, were not tested. Additionally, we used fixed frametime spike durations (0, 75, 150, 225 ms), which is decoupled from the frequency of frametime spikes that may be experienced in practice in real games. Moreover, our spikes are deliberately triggered by the action – this is to let us determine the effects of the spike on the action, compared to non-related spikes – whereas in practice spikes may be triggered by other system artifacts independent of the player action. While frametime spikes are rare in simple platformers, evaluating them provides baseline data for computationally-demanding games with similar mechanics, like jumping over gaps in Assassin's Creed. Furthermore, the randomized, movement-dependent timer caused exact spike locations to vary across participants, potentially introducing minor fluctuations in QoE.

We also used a single character with consistent movement, while more complex platformers often have varied mechanics and abilities. As a result, our findings may not apply to other navigation-based games with more diverse movement, like fighting games, rhythm games, or games with fluid, continuous steering such as racing.

7 Conclusion

Frametime spikes can occur in games due to various factors, such as loading events or network issues, affecting both the fluidity of gameplay and player performance. This study investigates the impact of frametime spikes on player performance and visual smoothness QoE during navigation-based tasks in a 2D platformer game. Using the open-source game SuperTux Classic, we designed and conducted a user study that isolates navigation tasks from other gameplay elements. Participants played custom levels of SuperTux, modified to throw frametime spikes at specific times during navigation actions.

A total of 31 participants played 32 rounds of the modified game, each round featuring a navigation-based task with fixed frametime spike sizes. Objective performance data (e.g., task completion) and subjective QoE data (e.g., perceived smoothness) were collected to assess the impact of these spikes.

The results indicated a clear negative correlation between frametime spike size and QoE across all task types, with larger spikes consistently reducing perceived smoothness. This impact on player performance was task-dependent. More difficult jumping-based tasks showed a significant decline in performance as spike size increased, while easier item collection and enemy squishing tasks did not show a clear correlation. This suggests that the impact

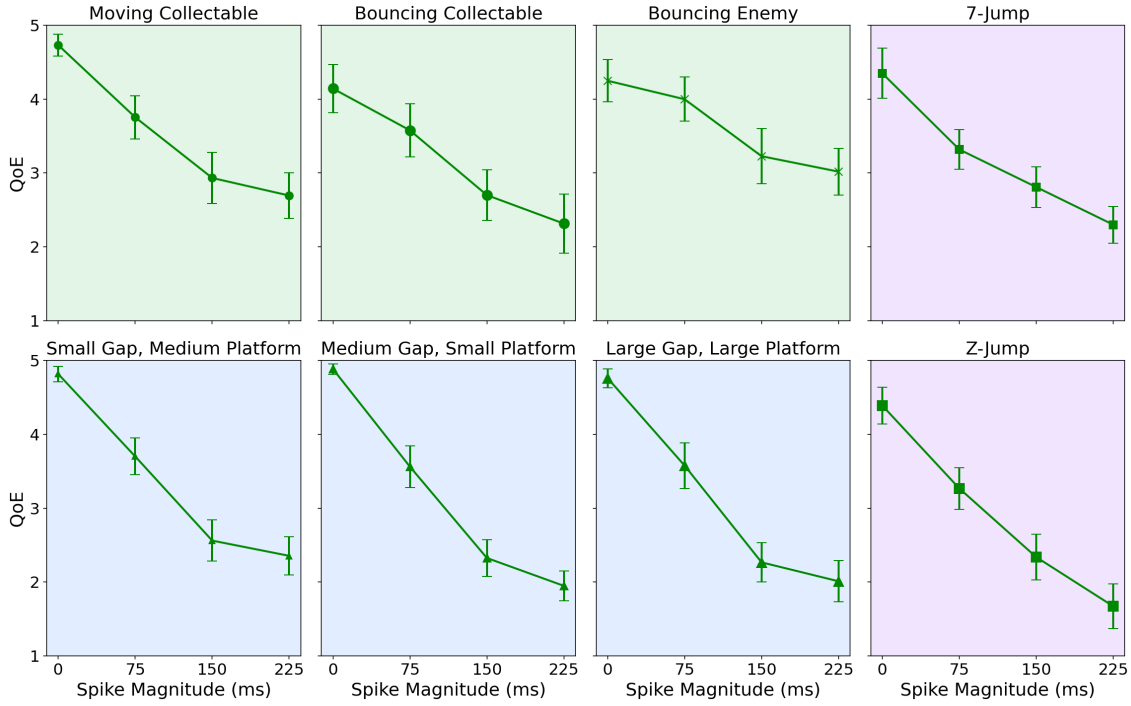


Figure 7: QoE vs. spike magnitude for each task. Mean with 95% confidence interval. Shaded areas indicate task type: green for interaction, blue for basic jumps, and purple for complex jumps.

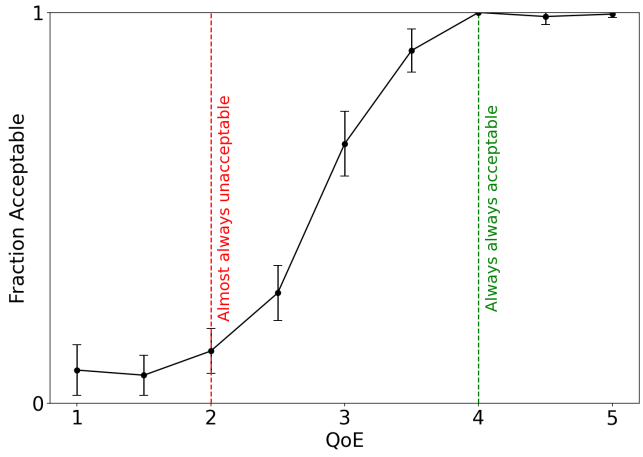


Figure 8: Fraction acceptable versus QoE. Mean with 95% confidence interval.

of frametime spikes is more pronounced in tasks requiring precise timing. These results can inform game developers and game system developers (e.g., cloud gaming platforms) of performance targets: since spike impact depends on when spikes occur, engines could defer non-critical computations (e.g., loading assets, garbage collection) away from high-impact actions like jumps.

Future work could focus on reducing the impact of frametime spikes by smoothing performance or adjusting gameplay mechanics, such as assisting players in a navigation action when a frametime spike is detected by the game engine. Other future work could study how spikes affect actions in other game genres, such as racing games, where timing and control during cornering could be disrupted. Another approach is to prevent spikes during important gameplay moments by separating out visually-disruptive system tasks during potentially high-impact actions like jumping a large gap, or to detect that spikes occurred and then look to mitigate them after the fact with game or system settings.

References

[1] Mohamed Abou-Zleikha and Noor Shaker. 2014. PaTux: An Authoring Tool for Level Design through Pattern Customisation Using Non-Negative Matrix Factorization. In *Proceedings of the 10th AAAI Conf. on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*. AAAI Press, 103–104. doi:10.1609/aiide.v10i1.12726

[2] E. Camilleri, G. N. Yannakakis, and A. Dingli. 2016. Platformer Level Design for Player Believability. In *2016 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, Santorini, Greece, 1–8. doi:10.1109/CIG.2016.7860404

[3] Kajal T. Claypool and Mark Claypool. 2007. On Frame Rate and Player Performance in First-Person Shooter Games. *Multimedia Systems* 13, 1 (2007), 3–17.

[4] Mark Claypool and Kajal T. Claypool. 2009. Perspectives, Frame Rates and Resolutions: It's All in the Game. In *International Conference on Foundations of Digital Games*. FL, USA.

[5] Mark Claypool, Kajal T. Claypool, and Feissal Damaa. 2006. The Effects of Frame Rate and Resolution on Users Playing First-Person Shooter Games. In *ACM/SPIE Multimedia Computing and Networking (MMCN) Conference*. San Jose, CA, USA.

[6] Marcello A. Gómez-Maureira, Michelle Westerlaken, Dirk P. Janssen, Stefano Gualeni, and Licia Calvi. 2014. Improving Level Design Through Game User Research: A Comparison of Methodologies. *Entertainment Computing* 5, 4 (2014), 463–473. doi:10.1016/j.entcom.2014.08.008

[7] Alyssa Ivanova. 2023. Revolutionizing Gameplay: Technological Advancements in Video Game Realms. https://tinyurl.com/4ypf3d9h. Accessed: Feb 19, 2025.

[8] Benjamin F. Janzen and Robert J. Teather. 2014. Is 60 FPS better than 30? The impact of frame rate and latency on moving target selection. In *CHI '14 Extended Abstracts on Human Factors in Computing Systems*. Association for Computing Machinery, Toronto, ON, Canada, 1477–1482. doi:10.1145/2559206.2581214

[9] Devi Klein, Josef Spjut, Ben Boudaoud, and Joohwan Kim. 2023. The Influence of Variable Frame Timing on First-Person Gaming. *arXiv preprint arXiv:2306.01691* (2023). cs.GR, arXiv:2306.01691.

[10] Shengmei Liu and Mark Claypool. 2022. The Impact of Latency on Navigation in a First-Person Perspective Game. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems (CHI '22)*. Association for Computing Machinery, New York, NY, USA. doi:10.1145/3491102.3517660

[11] Minzhao Lyu, Yifan Wang, and Vijay Sivaraman. 2024. Do Cloud Games Adapt to Client Settings and Network Conditions?. In *IFIP Networking Conference (IFIP Networking)*. 482–488. doi:10.23919/IFIPNetworking62109.2024.10619817

[12] Peter Quax, Patrick Monsieurs, Wim Lamotte, Danny De Vleeschauwer, and Natalie Degrande. 2004. Objective and subjective evaluation of the influence of small amounts of delay and jitter on a recent first person shooter game. In *Proc. 3rd ACM SIGCOMM Workshop on Network and System Support for Games (NetGames '04)*. ACM, 152–156.

[13] Henrique Souza Rossi, Niclas Ögren, Karan Mitra, Irina Cotanis, Christer Åhlund, and Per Johansson. 2022. Subjective Quality of Experience Assessment in Mobile Cloud Games, In GLOBECOM - IEEE Global Communications Conference. *IEEE Global Communications Conference (GLOBECOM)*, 1918–1923.

[14] Josef Spjut, Ben Boudaoud, Kamran Binaee, Jonghyun Kim, Alexander Majercik, Morgan McGuire, David Luebke, and Joohwan Kim. 2019. Latency of 30 ms Benefits First Person Targeting Tasks More Than Refresh Rate Above 60 Hz. In *SIGGRAPH Asia 2019 Technical Briefs* (Brisbane, QLD, Australia) *(SA '19)*. Association for Computing Machinery, New York, NY, USA, 110–113. doi:10.1145/3355088.3365170

[15] Mirko Suznjevic, Ivan Slivar, and Lea Skorin-Kapov. 2016. Analysis and QoE evaluation of cloud gaming service adaptation under different network conditions: The case of NVIDIA GeForce NOW, In Eighth International Conference on Quality of Multimedia Experience (QoMEX). *International Conference on Quality of Multimedia Experience (QoMEX)*, 1–6.

[16] Samin Shahriar Tokey, Ben Boudaoud, Joohwan Kim, Josef Spjut, and Mark Claypool. 2025. Pushing the Limits? Frame Rate Benefits to Players for up to 500 Hz in First Person Shooter Games. In *Proceedings of the 25th ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*. Stellenbosch, South Africa, 22–28. doi:10.1145/3712678.3721877

[17] Samin Shahriar Tokey, Ben Boudaoud, Joohwan Kim, Josef Spjut, and Mark Claypool. 2025. Timing Matters: The Impact of Event-Specific Frametime Spikes in First-Person Shooter Games. In *Proceedings of the 17th International Conference on Quality of Multimedia Experience (QoMEX)* (2025-09-30/2025-10-02). Madrid, Spain. http://www.cs.wpi.edu/~claypool/papers/frame-stutter-qomex-25/

[18] Samin Shahriar Tokey, Ben Boudaoud, Joohwan Kim, Josef Spjut, Peter Xenopoulos, and Mark Claypool. 2026. Impact of Graphical Fidelity and Frame-Time Stutter in a First-Person Shooter Game. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 9, 1, Article 16 (May 2026). doi:10.1145/3804493

[19] J. Wang, R. Shi, W. Zheng, W. Xie, D. Kao, and H.-N. Liang. 2023. Effect of Frame Rate on User Experience, Performance, and Simulator Sickness in Virtual Reality. *IEEE Transactions on Visualization and Computer Graphics* 29, 5 (May 2023), 2478–2488. doi:10.1109/TVCG.2023.3247057

[20] Xiaokun Xu and Mark Claypool. 2024. User Study-based Models of Game Player Quality of Experience with Frame Display Time Variation. In *15th ACM Multimedia Systems Conference*. Bari, Italy.

[21] Saman Zadtootaghaj, Steven Schmidt, and Sebastian Möller. 2018. Modeling Gaming QoE: Towards the Impact of Frame Rate and Bit Rate on Cloud Gaming. In *Conference on Quality of Multimedia Experience (QoMEX)*. Cagliari, Italy.