Timing Matters: The Impact of Event-Specific Frametime Spikes in First-Person Shooter Games

Samin Shahriar Tokey*, Ben Boudaoud†, Joohwan Kim†, Josef Spjut†, Mark Claypool*
*Worcester Polytechnic Institute, USA
†NVIDIA, USA
{sstokey, claypool}@wpi.edu, {bboudaoud, jspjut, sckim}@nvidia.com

***Abstract*—Frametime spikes can disrupt gameplay in first-person shooter (FPS) games, affecting both performance and player experience. This paper examines how spikes during specific game events impact players. We developed a custom FPS game that maintains a steady 500 frames/s while inducing frametime spikes during weapon reloading, fast mouse movement, or targeting. Thirty-eight (38) participants played the game in a user study, providing both performance data and user-reported visual smoothness. Results show that spikes while targeting lowered accuracy and score, while spikes during reloads and mouse movement did not affect performance but still degraded user experience. These results suggest that both the relative timing and size of frametime spikes matter in FPS gameplay. Per-action models better account for average QoE when spikes happen, showing better fit than models that only consider spike size (independent of action).**

***Index Terms*—First-person Shooter, FPS, High Refresh Rate, Frametime variation, Framerate, Stutters, Lag, 500 Hz, QoE, Quality of Experience, Visual Perception**

I. Introduction

Games continue to influence numerous advances in computer technologies, with developments in hardware and software improving both player experience and performance [1]. In first-person shooter (FPS) games, precision and responsiveness are essential, and high frame rates are especially important for competitive esports players [2]. However, a high average frame rate alone is insufficient as fluctuations in frame rate can disrupt gameplay and degrade the overall experience [3]. Achieving smooth framerates with minimal variation in frametimes is challenging, especially as the demand for high framerate gaming increases.

Previous studies have looked at the effects of frametime variation on player performance and experience [4]–[7]. Frametime *spikes* or *stutters* – large deviations from target frametimes – can degrade player experience and performance. Although the effects of frametime spikes in general have been studied, the effects of frametime spikes that occur as a player tries to complete a task remain unexplored.

Our study focuses on the effects of frametime spikes during specific game events in a FPS game, in contrast to earlier work that used random or periodic stutters that were independent of game events. We design a user study with a custom FPS game called *Lead Rush* that included core FPS gameplay elements such as shooting, reloading, aiming down sight, walking, and strafing. We selected three game events, each with a different expected level of impact on player performance and quality of experience: weapon reloading, rapid mouse movement, and on-target aiming. Frametime spikes were triggered by the player action, with nanosecond precision, and their magnitudes were varied between 75 ms, 225 ms, and 675 ms. Participants played Lead Rush for about 20 short game rounds that had different frametime spike settings while recording performance metrics (e.g., score, accuracy) and user perception of visual smoothness. This approach allows analysis of the impact of frametime spikes during game events on both player performance and quality of experience (QoE). We also used the data to model player QoE, including comparison to previously derived models [8].

The rest of this paper is organized as follows: Section II summarizes related research; Section III outlines our methodology – assessing the effects of frametime spikes on different game events through a user study; Section IV analyzes the results from the user study; Section V discusses our key findings and their implications; Section VI identifies some study limitations; and Section VII summarizes our conclusions and suggests possible future work.

II. Related Work

A. Frametime Variation and Adaptive Displays

Since player input relies on real-time visual cues, especially in fast-paced FPS games, framerates and frametime variability significantly impact player experience and performance. Klein et al. [4] studied variable frame timing (VFT), where frame intervals fluctuate. They found that large deviations affected motion smoothness but had minimal impact beyond perception.

Liu et al. [3] examined quality of experience by artificially increasing CPU load to induce frametime variations in the FPS game *Valorant*. They observed that greater frametime variation reduced QoE. Tokey et al. [9] used high refresh rate displays to evaluate the effects of framerate up to 500 frame/s on FPS games. They found that player performance stopped improving after 60 frame/s, but players continued to perceive increased smoothness up to 500 frame/s.

Technologies like VSync and G-Sync are designed to help mitigate framerate variation by (dynamically) synchronizing refresh rates with frame rates. Lee et al. [5] found that higher server tick-rates and VSync improved player accuracy in FPS games. Watson et al. [6] found G-Sync improved player performance in *Battlefield 4*, an FPS game. Denes et al. [7]

optimized motion quality by adjusting refresh rates based on visual target speed. Their tests of 50-165 Hz displays showed adaptive refresh rates provided smoother visuals than fixed refresh rates, benefiting G-Sync monitors and VR/AR headsets. Although these techniques help with variation in framerate, they cannot mitigate the effects of large frametime spikes.

B. Frametime Variation and Cloud Games

Some studies have examined how network jitter – which affects frametime variation – affects QoE in cloud gaming. Rossi et al. [10] studied mobile cloud gaming on smartphones, finding that game streams react to network perturbations compared to traditional online games, which can mean increased frametime variation. Suznjevic et al. [11] evaluated GeForce Now's adaptation algorithm under varying network conditions, showing that added jitter can force the system to drop frame rates and resolutions to their lowest supported levels.

Xu et al. [8] built a QoE model from four user studies across 11 games, considering framerates and frametime variations. They confirmed that while average framerate generally correlated with QoE, it did not always predict satisfaction accurately. They found that frametime variation and interruption severity were strong predictors of player experience, explaining about 90% of gaming quality variations. Additionally, interruption frequency had little effect on experience, and models focusing on the lowest framerates failed to consistently capture player satisfaction across different games and conditions. Similar findings were reported for VR cloud FPS games, where network jitter and RTT were found to be key predictors of QoE [12].

Our study examines the impact of large frametime variations, known as stutters or spikes, in a FPS game. Unlike past research that studied frametime spikes that occurred independently of in-game events, we focus on the effects of frametime spikes that are triggered by specific in-game events corresponding to player actions. We also develop a QoE model that accounts for these event-specific frametime spikes and compare it to general models, similar to those developed in a previous study.

III. Methodology

To investigate the effects of event-based frametime spikes in a FPS game, we conducted a user study using our custom game *Lead Rush*.

A. Game Description

Lead Rush[1] is a FPS game developed in Unity 2023 to run at ultra-high frame rates (1500 frame/s). It features procedural animations, positional audio, and a configurable experimental harness for data collection, extended with a logging system to record performance metrics.

Figure 1 shows a screenshot of *Lead Rush*. Hit markers appear on successful hits (yellow) and enemy destructions (red), though these are not shown in the figure.

The game involves controlling a first-person camera view to shoot and destroy a single enemy at a time while avoiding

Fig. 1: Lead Rush screenshot. The enemy is the red orb in front of the player, indicated by the yellow arc at the top.

contact with the enemy. Enemies are red floating orbs that require 5 hits to destroy. The orb moves toward the player, navigating around obstacles. If the orb collides with the player avatar, both are eliminated, and the player avatar respawns at the starting position. After an orb is destroyed, a new one respawns in 100 milliseconds. The orb's respawn location is dynamically set within a torus-shaped area, 4 to 6 Unity meters from the player, avoiding blocked areas. The orb oscillates vertically between 0.9 and 1.1 Unity meters above ground (once per second) and moves at 3.1 meters per second, slightly faster than the player avatar's 3 meters per second. A Unity Navmesh[2] is used navigation.

The game provides visual and audio cues for enemy direction and proximity. Yellow arcs flash along the screen edges, growing brighter as the enemy gets closer. Positional audio indicates enemy location, with distinct sounds for hits, destruction, and shooting.

Although Lead Rush has various types of weapons typically found in commercial FPS games, our user study restricted players to only have a fully automatic (hold click to fire) weapon with a 750 rounds/minute fire rate. The game detects hits via hitscan, not propagated projectiles, and provides a green laser pointer to assist the player with aiming. The weapon has procedural recoil, sway, camera shake and aim-down sight animations. A yellow hit marker appears in the center of the screen after a successful hit, and a red marker is displayed when the player destroys an enemy. The weapon has unlimited ammo, with 31 shots before a reload is required. The weapon automatically reloads when the ammo is empty, but the player can also press 'R' to perform an on-demand reload. The reload animation takes about 2 seconds to complete.

Players earn 10 points per hit and 100 points per kill but lose 100 points per avatar death. Each missed shot deducts 1 point. Scores can drop below zero.

B. Modifications for Experiments

For our study, Lead Rush was instrumented to introduce controlled frametime spikes during several game events. To throw a frametime spike of a fixed size, the game calculates the duration of the main loop and then runs a busy loop for the difference between the set magnitude and the main loop

[1]Lead Rush Repository: https://github.com/Tokey/Lead-Rush

[2]Navmesh: https://docs.unity3d.com/ScriptReference/AI.NavMesh.html

time. The timing is measured with nanosecond precision. A test harness records the frametime spikes during specific game events, as well as recording objective player performance and subjective QoE. For our study, we used 3 spike events with a fixed magnitude thrown during specific game actions/events common to FPS games, each with a different expected impact from the spikes:

1) *Reload (expected low impact).* When enabled, a frametime spike occurs when a weapon reload is initiated, either automatically when out of ammo or manually when the user presses 'R'.
2) *Mouse Speed (expected medium impact).* When enabled, a frametime spike is triggered when the mouse is moved rapidly (say, when quickly rotating the avatar's view to find the enemy orb), with a spike threshold of 1.3 degrees per frame, determined through a pilot study. The cooldown between spikes is 0.75 seconds.
3) *On Target (expected high impact).* When enabled, a frametime spike is triggered when the player's reticle gets close to targeting an enemy orb. An invisible collider surrounding the orb oscillates between 2.5 and 3.5 Unity units every 2 seconds, introducing randomness to the spike timing. A spike occurs when the reticle intersects with the invisible collider, with no additional spike until the reticle exits the collider. The cooldown between spikes is 0.75 seconds, determined through a pilot study.

The spike magnitudes were 0 ms, 75 ms, 225 ms, and 675 ms, selected based on pilot tests (75 ms was "just noticeable", 675 ms was "just playable" and 225 ms was and in-between "medium"). The framerate was set to 500 frame/s for all rounds. For the 0 ms spike, no extra delay was added, so the inter-frame time stayed around 2 ms. For the other conditions, additional delay was added on top of the base 2 ms to create the desired spike (e.g., 675 ms spike included an added delay of 673 ms). Each participant completed 20 rounds, consisting of practice, main, and control rounds as summarized in Table I.

Round Type	**Details**
Practice Rounds	2 rounds (0 ms, 675 ms on-target)
Main Rounds	3 mouse speed spike rounds (75/225/675 ms)
	6 reload/on-target spike rounds
Control Rounds	3 rounds without spikes (Rounds 3, 11, 20)
Total Rounds	20 (each magnitude repeated twice for main rounds)

TABLE I: User study round configuration

A session ID was used to track each row of the Latin square. After a user completed a session, the session ID was incremented and saved to a file. For a new session, the ID was read from the file, and the corresponding row of the Latin square was used to determine the sequence of rounds.

The game generated logs to track player actions, performance metrics, and enemy interactions, which were used in our analysis.

After the round ended, players were asked a question regarding their experience for that round:

- *Rate the visual smoothness of the round* (Very Choppy) 1 to 5 (Very Smooth), via a slider.
- *Was the smoothness of the round acceptable?* Yes / No, via radio buttons.

C. User Study Procedure

The user study procedure was approved by our university's institute review board and was as follows:

1) *Recruiting:* Students were recruited via relevant mailing lists and an official Discord play testing channel, with interested participants selecting their preferred time slots using Slottr.[3]
2) *Informed consent and demographics:* Before starting, each participant was given a consent form that informed them about the content of the game and study procedure, and then completed a demographic survey (all questions optional).
3) *Reaction time test:* Each participant did a reaction time test similar to the Human Benchmark Reaction Time Test[4] [13].
4) *Practice rounds:* Participants started by playing two practice during which they were informed about the QoE question presented at the end of each round. The first round had no spike, and the second round had a 675 ms on-target spike. These were shown as the best and worst cases to give participants clear reference points. This provided a baseline for the full range of the QoE slider instead of evolving ratings based on the order in which spikes were encountered.
5) *Main rounds:* After the practice rounds, participants completed the 15 main rounds and 3 control rounds, shuffled via a latin square.
6) *Collection:* Log files from each session were collected and backed up by the study proctor.

Each session lasted about 30 minutes. Participants were entered into a raffle for a $15 USD Amazon eGift card and the highest scorer received the same as a reward.

D. Hardware

System specifications are provided in Table II.

The duration of the frametime spikes thrown was recorded by the game's log system. For the 75 ms, 225 ms and 675 ms spikes, all three conditions show means of 75.0 ms, 225.0 ms, and 675.0 ms, respectively. Standard deviations range from 0.05 - 104 ns. Minimum and maximum values are close to the target values – differences are 0.007% for reload 75 ms and approximately 0% for the the others.

IV. Analysis

This section summarizes participant demographics and provides an analysis of player experience and performance. The complete dataset, including all session folders, is available online for reference and analysis.[5]

[3]Slottr: https://www.slottr.com/
[4]Reaction Time Test: https://humanbenchmark.com/tests/reactiontime
[5]Complete dataset: https://tinyurl.com/FTLeadRush

CPU	Intel Core i9-11900K
GPU	NVIDIA RTX 4070 Super FE
RAM	32 GB DDR4
Storage	Samsung 970 EVO SSD
Mouse	Logitech G502
Keyboard	Logitech G910
Display	Alienware AW2524H (500 Hz, 1920×1080)
Latency	<20 ms (at 500 Hz)
PC Latency	$\mu = 5.1$ ms, $\sigma = 10.3$ ms
System Latency	$\mu = 7.6$ ms, $\sigma = 10.3$ ms

TABLE II: System for user study. Data was collected using NVIDIA GeForce Experience (v3.28.0.412 - Experimental).

TABLE III: Participant demographics.

Users	**Age**	**Gender**	**Gaming Skill** (1-5)	**FPS Skill** (1-5)	**Reaction Time** (ms)
38	20.4 (2.7)	♂30 ♀5 ⚧3	3.5 (0.7)	3.0 (0.8)	194.5 (78.9)

A. Demographics

A total of 38 players participated in the study. Table III summarizes their demographics. Values are means with standard deviations in parentheses. The average participant age was approximately 20 years old (μ = 20.4, σ = 2.7), ranging from 18 to 24 years old, with the majority identifying as male (79%). Participants rated their experience in general games and FPS games with a mouse and keyboard on a scale from 1 (low) to 5 (high). On average, participants had moderate gaming experience (μ = 3.5, σ = 0.7) and moderate FPS experience (μ = 3.0, σ = 0.8). Average reaction times were quick (μ = 194.5 ms, σ = 78.9), typical of experienced gamers [14].

B. Player Performance

This subsection first analyzes player score, which gives a broad view of player performance, followed by accuracy which provides a more low-level measure of performance.

Table IV shows mean frametime spike count taken across all users and all rounds, with min, max and standard deviation.

TABLE IV: Frametime spike count statistics per round.

Event	**Mean**	**Min**	**Max**	**Stddev.**
Reload	5.8	2.0	12.0	1.6
Mouse speed	7.6	0.0	25.0	5.6
On target	36.6	2.0	52.0	5.4

1) Score: Figure 2 shows the average player scores for all users averaged across all rounds for different conditions. The y-axis is the score and the x-axis is the frametime spike magnitude, with data colored by action. In the graph, player performance appears unaffected by 75 ms frametime spikes for any action. However, for frametime spikes during targeting, performance declines at both 225 ms and even more at 675 ms, while frametime spikes during the other actions do not have a similar effect.

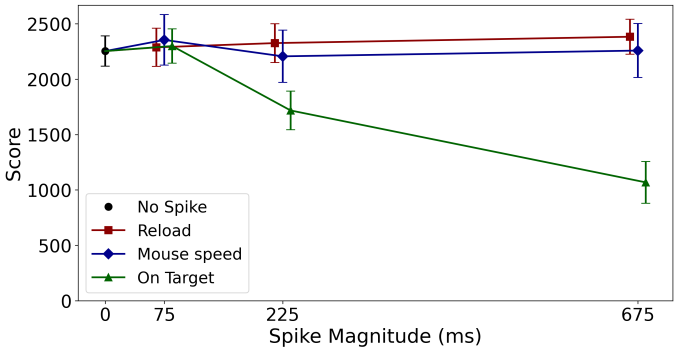

Fig. 2: Score for frametime spikes during different actions. Values are means with 95% confidence intervals as error bars.

2) Accuracy: Figure 3 shows the accuracy vs. spike magnitude with the same statistics as in Figure 2, but with the y-axis as shots hit divided by shots fired, as a percent. This graph demonstrates similar trends for accuracy as for score, where frametime spikes during targeting reduce player accuracy at 225 ms and 675 ms, while spikes during other actions show little impact on accuracy.

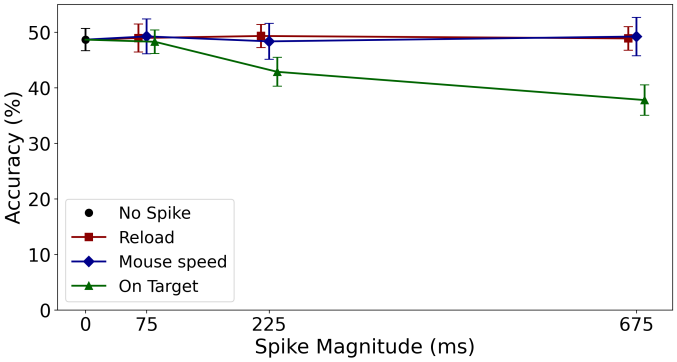

Fig. 3: Accuracy for frametime spikes during different actions. Values are means with 95% confidence intervals as error bars.

C. Player Experience

This subsection analyzes the effects of frametime spikes and action on player-reported QoE (visual smoothness) and acceptability at the end of each round. A QoE model was developed using our data and compared to a model derived from previous user studies.

1) Quality of Experience (QoE): At the end of each round, users provided an assessment of visual smoothness (1-low to 5-high), which we call QoE. Figure 4 shows the QoE analysis, with the same statistics as in Figure 2 but for player-reported QoE. From the graph, players reported degraded QoE for frametime spikes during targeting at 75 ms and for frametime spikes of 225 ms and 675 ms during all actions. This suggests that although frametime spikes triggered by mouse speed and weapon reload do not have much impact on player performance (see Figures 2 and 3), these spikes are still noticeable and

degrade QoE, although less so than spikes that happen at more sensitive times, like when targeting.

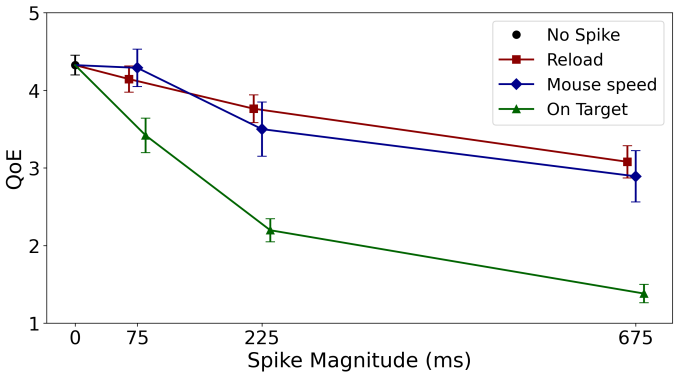

Fig. 4: QoE for frametime spikes for different actions. Values are means with 95% confidence intervals as error bars.

2) Acceptability: Figure 5 shows the relationship between users' acceptability ratings and their corresponding QoE scores. The x-axis is the QoE rating (in 0.5 unit bins) and the y-axis is the percent that users found rounds with that QoE acceptable. Each point is the mean value averaged across all rounds shown with a 95% confidence interval. From the graph, rounds with a QoE above 4 were always acceptable, whereas those with a QoE below 2 were nearly always unacceptable, with a sharp change from unacceptable to acceptable in-between.

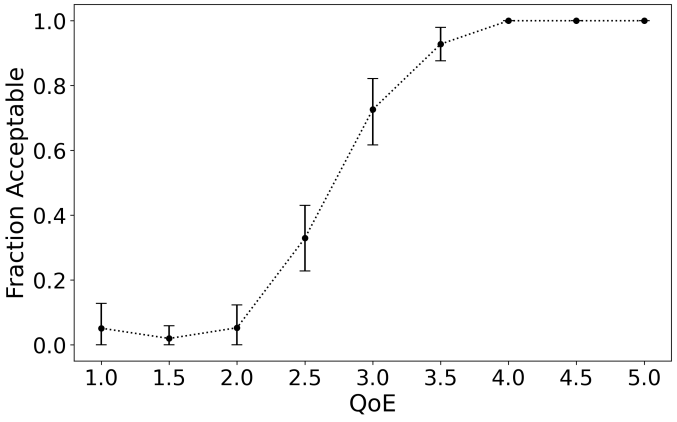

Fig. 5: Fraction acceptable versus QoE.

The acceptability versus QoE relationship appears sigmoidal, and can be modeled with the function:

$$f(x) = \frac{1}{1 + e^{-k(x - x_0)}} \tag{1}$$

where k controls the slope and x_0 is the midpoint. The sigmoid model has a steep slope, with k = 3.47 and a high R^2 of 0.998.

Using this function and the reported QoE scores, Figure 6 shows a mapping of acceptability replacing the y-axis (QoE) in Figure 4. From the graph, frametime spikes of 75 ms are generally always acceptable, except when they occur on targeting where they are unacceptable about 10% of the time. For frametime spikes of 225 ms during targeting, acceptability drops to about 20% and degrades for all actions by about 10%. Frametime spikes during targeting at 675 ms are almost never acceptable, and even when they happen during other actions, degrade acceptability by about 25%.

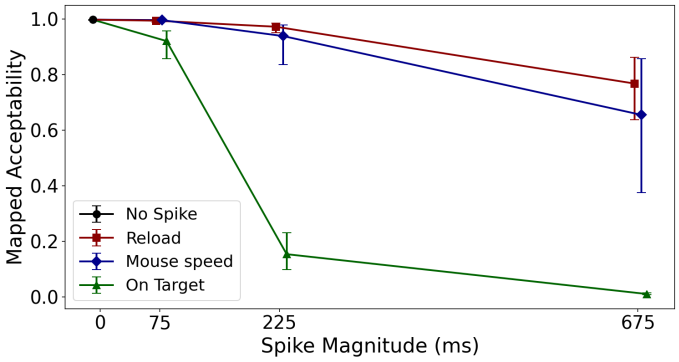

Fig. 6: Mapped acceptability versus frametime spikes for different actions. Values are means with 95% confidence intervals as error bars.

3) QoE Modeling: We observed that QoE trends in our study are non-linear and strongly depend on the specific action during which the frametime spike occurs. We therefore developed separate exponential models of average QoE for each action – reload, mouse speed, and on target – using spike magnitude as the independent variable. Our models are given by:

$$\text{QoE} = a \cdot e^{bx} \tag{2}$$

where x denotes the spike magnitude and a and b are parameters derived from our data.

For spike magnitude, the per-action exponential models fit the mean QoE values well, with R^2 values of 0.99 for reload, 0.93 for mouse speed, and 0.91 for on-targeting. In contrast, a combined exponential model that pools all actions performs much worse, with an R^2 of 0.51. We also fit the linear model from Xu and Claypool [8] that predicted QoE for random frametime variation, but found that it was not a good fit for our data ($R^2 = -0.01$ where slope = -0.004 and intercept = 4.0).

These values are summarized in Table V.

TABLE V: Exponential models of QoE (smoothness) for spike magnitude.

Condition	a	b	R^2
Reload	4.26	-0.00050	0.99
Mouse speed	4.25	-0.00057	0.93
On target	3.83	-0.00159	0.91
All (combined)	4.02	-0.00084	0.51

V. Discussion

Frametime spikes during high-impact events, such as targeting, affect both performance and player experience. Spikes

while targeting had the strongest impact, lowering accuracy, score, and QoE as magnitude increased. In contrast, spikes during reloads or rapid mouse movement affected QoE but had little impact on performance. This suggests that while players noticed the visual stutters no matter when they occur, they matter less when they do not interfere with the primary task of targeting. Together, these results confirm that the impact of frametime spikes depends upon when they occur, with spikes that happen during critical actions (e.g., targeting) impacting performance and QoE much more than spikes during other actions (e.g., reload). This may be expected for player performance, but our results also demonstrate this holds for player perception of visual smoothness – i.e., players do not notice as much degradation to visual smoothness for the same-sized spike during a non-critical action versus a critical action.

Since spike-impact depends upon player action, our per-action QoE models capture these effects better than models derived for random spikes over all actions. This suggests that monitoring spike magnitudes independently of player action does not as effectively predict QoE as does monitoring when the spike occurs relative to the player action. Correspondingly, treatments in the game or game system that seek to ameliorate the impact of frametime spikes could take corrective action based on the expected player intent (e.g., assisting with aiming when a frametime spike is detected on a targeting event).

VI. Limitations

Our study focused on frametime spikes during three specific in-game player actions: reloading, fast mouse movement, and targeting an enemy. While these represent core actions in FPS games, many other in-game actions, such as weapon switching, jumping and strafing, or interacting with environment elements (e.g., picking up ammo) could also be affected by frametime spikes. According to our preliminary testing, frametime spikes in commercial FPS games are often encountered during loading processes, such as scene transitions, texture streaming, or asset loading. These spikes may not always happen during high-impact gameplay events but can still degrade the quality of experience.

We designed our study around the core mechanics of an FPS game – aiming, shooting, and movement in combat with a single enemy. Most commercial FPS games include additional mechanics, such as multiple weapon types, tactical abilities, and dynamic environments, which could influence how players perceive and adapt to frametime spikes. Our study used a single fully-automatic weapon, but different weapon types, particularly those requiring precise aiming, may be more sensitive to frametime spikes. Also, our enemy did not shoot back – it only moved toward the player before exploding – unlike opponents in may FPS games. In a scenario with multiple enemies firing at the player, rapid mouse movement would be used more often, thus potentially shifting the impact for frametime spikes, for example, compared to our results.

Additionally, our study focused only on FPS games. Other genres, such as third-person shooters, racing games, platformers, or strategy games, may experience frametime spikes differently. However, our findings do suggest player action and game task are relevant to understanding and predicting the effect of frametime spikes, and games with actions with similar precision and deadline to those in our game could show similar effects.

Our study used 500 frames/s because it reflects current state-of-the-art PC performance, which makes the study more future-proof as higher framerates become more common. It is possible that the degradation to visual smoothness may differ for lower frame rates.

VII. Conclusion

Frametime spikes can occur in games due to level loading, system processes, or network issues, affecting how smooth a game feels and how well players perform. This study examines how these spikes impact player experience and performance in a first-person shooter (FPS) game when they occur during specific game events. Using a custom FPS game, *Lead Rush*, we designed and conducted a user study assessing the effects of frametime spikes during specific gameplay moments while keeping experimental conditions consistent across independent variables.

A total of 38 participants each played 20 one-minute rounds of Lead Rush at a fixed 500 frame/s. Frametime spikes were introduced during three different in-game events – reloading, fast mouse movement, and targeting – at three frametime spike magnitudes (75 ms, 225 ms, and 675 ms). Objective data on player performance (e.g., score, accuracy) and subjective data on player QoE (e.g., perceived smoothness, acceptability) were collected to assess the impact of these spikes.

The results show that frametime spikes while targeting reduce accuracy and score, with stronger effects as spike magnitude increases. Frametime spikes during weapon reloading and rapid mouse movement did not impact player performance but still degraded QoE, indicated players felt the spikes degraded their experience, albeit less so, even if their gameplay did not suffer. This shows that *when* a frametime spike happens during a game matters, not just how severe it is, and that the same sized spike during a non-critical action not only may not affect performance, but may not matter to the player experience as much as a spike during a critical action. Our per-action QoE models capture these effects better than models derived for action-independent frametime spikes.

Future work could focus on reducing the impact of frametime spikes by smoothing performance or adjusting gameplay mechanics, such as increasing hitbox sizes to ameliorate the effects frametime spikes when targeting. 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, or platformers, where precise jumps and mid-air adjustments may be affected. Another approach could try to prevent frametime spikes during important gameplay moments by separating out visually-disruptive system tasks during high-impact actions like aiming and shooting.

REFERENCES

[1] A. Ivanova, "Revolutionizing Gameplay: Technological Advancements in Video Game Realms," https://tinyurl.com/4ypf3d9h, Aug 2023, accessed: Feb 19, 2025.

[2] T. Tamasi, "Why Does High FPS Matter For Esports?" https://www.nvidia.com/en-us/geforce/news/what-is-fps-and-how-it-helps-you-win-games/, Dec 2019, accessed: Feb 19, 2025.

[3] S. Liu, A. Kuwahara, J. Scovell, J. Sherman, and M. Claypool, "The Effects of Frame Rate Variation on Game Player Quality of Experience," in *ACM CHI Conference on Human Factors in Computing Systems*, Hamburg, Germany, Apr. 2023.

[4] D. Klein, J. Spjut, B. Boudaoud, and J. Kim, "The Influence of Variable Frame Timing on First-Person Gaming," *arXiv*, vol. abs/2306.01691, 2023.

[5] W. K. Lee and R. Chang, "Evaluation of Lag-related Configurations in First-Person Shooter Games," in *Proceedings of ACM/IEEE Network and System Support for Games (NetGames)*, Delft, the Netherlands, 2015.

[6] B. Watson, A. Spaulding, T. Davis, Y. He, K. Yao, and A. King, "The Effects of Adaptive Synchronization on Performance and Experience in Gameplay," *ACM Computer Graphics and Interactive Techniques*, vol. 2, no. 2, pp. 1–13, 2019.

[7] G. Denes, A. Jindal, A. Mikhailiuk, and R. K. Mantiuk, "A Perceptual Model of Motion Quality for Rendering with Adaptive Refresh-rate and Resolution," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 4, 2020.

[8] X. Xu and M. Claypool, "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, Oct. 2024.

[9] S. S. Tokey, B. Boudaoud, J. Kim, J. Spjut, and M. Claypool, "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, 2025, p. 22–28.

[10] H. S. Rossi, N. Ögren, K. Mitra, I. Cotanis, C. Åhlund, and P. Johansson, "Subjective Quality of Experience Assessment in Mobile Cloud Games," in *Proceedings of the IEEE Global Communications Conference (GLOBECOM)*, 2022, pp. 1918–1923.

[11] M. Suznjevic, I. Slivar, and L. Skorin-Kapov, "Analysis and QoE Evaluation of Cloud Gaming Service Adaptation under Different Network Conditions: The Case of NVIDIA GeForce NOW," in *Proceedings of the Eighth International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE, 2016, pp. 1–6.

[12] H. S. Rossi, K. Mitra, C. Åhlund, and I. Cotanis, "QoE Models for Virtual Reality Cloud-based First Person Shooter Game over Mobile Networks," in *2024 20th International Conference on Network and Service Management (CNSM)*, Prague, Czech Republic, 2024, pp. 1–5.

[13] X. Xu and M. Claypool, "Reflex - An Open-source Tool for Measuring Human Reaction Times," Computer Science Department at Worcester Polytechnic Institute, Tech. Rep. WPI-CS-TR-25-02, 2025.

[14] C. Jiang, A. Kundu, and M. Claypool, "Game Player Response Times versus Task Dexterity and Decision Complexity," in *Extended Abstracts of the ACM Annual Symposium on Computer-Human Interaction in Play (CHI Play)*, Virtual Event, 2020, p. 277–281.