The Effects of Network Latency on the Peeker's Advantage in First-person Shooter Games

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

Zesheng Chen
zchen13@wpi.edu
Worcester Polytechnic Institute
Worcester, MA, USA

Colin Mettler
csmettler@wpi.edu
Worcester Polytechnic Institute
Worcester, MA, USA

Dexuan Tang
dtang2@wpi.edu
Worcester Polytechnic Institute
Worcester, MA, USA

Ben Boudaoud
bboudaoud@nvidia.com
NVIDIA
Durham, NC, USA

Joohwan Kim
sckim@nvidia.com
NVIDIA
Santa Clara, CA, USA

Josef Spjut
jspjut@nvidia.com
NVIDIA
Durham, NC, USA

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

ABSTRACT

In first-person shooter (FPS) games, the *peeker's advantage* is the edge the moving peeker gets when battling a stationary defender at a corner due to network latency. However, confirmation of (the size of) this advantage based on network latency and the distance from the corner has not been studied. This paper assesses the peeker's advantage via two user studies both using an open-source FPS game extended to support two-player networking and a custom map. Users play as both peeker and defender with 3 different corner distances and 3 different network latencies. Analysis of hits, wins, and time-to-damage shows that the advantage for the peeker is impacted more by the defender's latency than the peeker's latency and is lowest when the peeker is nearest the corner. The user study with a tournament setting had quicker and more competitive matches resulting in more combat encounters and closer games than did the traditional user study.

CCS CONCEPTS

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

KEYWORDS

First Person Shooter, Latency, Peeker's Advantage, First Person Science, Networked Game

ACM Reference Format:
Samin Shahriar Tokey, Zesheng Chen, Colin Mettler, Dexuan Tang, Ben Boudaoud, Joohwan Kim, Josef Spjut, and Mark Claypool. 2024. The Effects of Network Latency on the Peeker's Advantage in First-person Shooter Games. In *Proceedings of the 19th International Conference on the Foundations of Digital Games (FDG 2024), May 21–24, 2024, Worcester, MA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3649921.3650002

1 INTRODUCTION

The global shooting games market, encompassing genres such as first-person shooters, is experiencing significant growth. The market is valued at USD 8.8 billion in 2020 and is forecasted to expand at a compound annual growth rate (CAGR) of 8.8% from 2020 to 2027. This robust growth trajectory is expected to culminate in a market value of USD 16.8 billion by the end of 2027 [28].

In FPS games, peeking around a corner and moving to attack a stationary defender is a core gameplay mechanic. A *peeker's advantage* arises in part due to the networked architecture multi-player FPS games use, wherein when a player interacts with their local game world, their client must relay these actions via a server to the other client(s) before other player(s) see them. Consider the example in Figure 1 showing an event timeline for a networked game with two clients and a server, with time progressing top to bottom. The left side shows the display on the client over this time. Player 1's action to move their avatar on client 1 is sent to the server and then, after processing, to client 2 where the new world is rendered for player 2. The latency between client 1 and the server and the server and client 2 limit how soon player 2 sees actions taken by player 1, thus providing an advantage for the peeker (player 1) moving to take a corner from a stationary defender.

Figure 2 depicts a top-down view of the peeker's advantage. At time $t = 1$ (a), the defender is holding a corner and the peeker is preparing to move into position to be able to shoot the defender. At time $t = 2$ (b), the peeker has moved into position and can see and shoot the defender, but because of network latency, the position of the peeker on the defender's display lags behind and the defender is still aiming. As a result, at time $t = 3$ (c), the peeker is able to eliminate the defender's avatar before the defender gets off a shot. In fact, with a fast and skilled enough peeker and/or a high enough

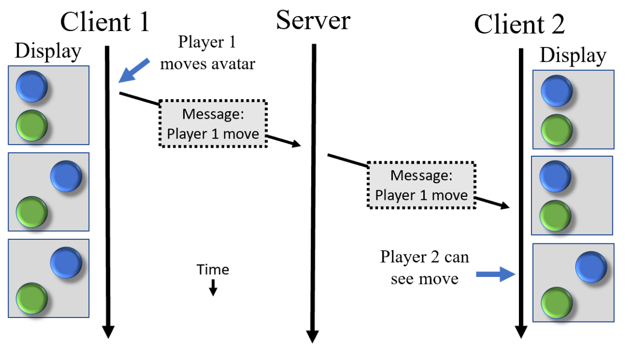


Figure 1: Network latency's effects on display time of actions taken by another player in a multi-player game.

latency, a defender can be eliminated before the peeker is even rendered on the defender's display.

Figure 3 provides a detailed timeline illustrating the peeker's advantage. The peeker, server and defender timelines are depicted advancing from left to right. At the top, the peeker moves around a corner revealing the defender. That movement then propagates first to the server and then to the defender whereupon the defender can see the peeker also. Meanwhile, the peeker has been able to aim and shoot at the defender, that shot needing to travel to the authoritative server before it is validated. The defender can also aim and shoot at the peeker but because of the network latency, the time to do so is less than that of the peeker. Once the server resolves the hit (in this example, the peeker has shot, hit and eliminated the defender and the game respawns both avatars), the server sends notice to respawn the avatars, as needed, setting up the next encounter for both the peeker and defender. A hit message arriving at the server after this respawn message (e.g., in this example, the defender's hit message) is likely to be resolved by the server as invalid – i.e., the peeker shoot first and the defender's shot is too late.

Independently of latency, the amount of advantage a peeker may have depends upon the relative distance of the peeker and the defender to the corner, depicted in Figure 4. On the left, the peeker loses some advantage when the defender's avatar is further from the corner than the peeker's, as the geometry causes more of the peeker's avatar to protrude and become visible to the defender. Consequently, the peeker can see less of the defender's avatar and, in fact, the defender can see and potentially shoot the peeker before even being seen. On the right, the opposite is true when the peeker is further away from the corner than the defender, where the peeker has both an earlier and larger target than the defender. However, the peeker in positioning their avatar further from the corner has made the defender target quite small and difficult to hit, possibly negating the advantage.

Despite the prevalence of peeker-defender encounters in FPS games and general awareness of the peeker's advantage by competitive FPS players (e.g., [1]), there are few studies measuring its effects. Doing so can not only confirm general understanding of player performance during a common tactical operation, but may yield insights into how experts (e.g., competitive FPS gamers) use computer interfaces. Our paper seeks to address this by isolating

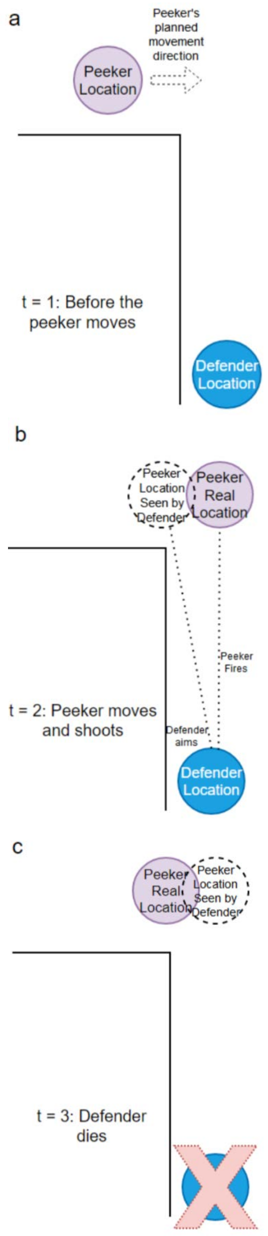

a
Peeker's planned movement direction
Peeker Location
t = 1: Before the peeker moves
Defender Location
b
Peeker Location Seen by Defender
Peeker Real Location
Peeker Fires
t = 2: Peeker moves and shoots
Defender aims
Defender Location
c
Peeker Real Location
Peeker Location Seen by Defender
t = 3: Defender dies

Figure 2: Timeline depicting peeker moving and shooting a defender holding a corner.

the roles of a peeker and defender in an FPS game while controlling peeker distance and network latencies in a user study. We extended the First-Person Science open-source shooter game [5] to support two networked players and designed a custom map that forced players into either a peeker or defender role. For the study, users were randomly assigned a role (peeker or defender), one-way latency to the server (0 ms, 30 ms, 60 ms), and peeker distance from the corner (7 m, 19 m, 27 m) and then competed for a round (15 seconds), repeating for all combinations for each player.

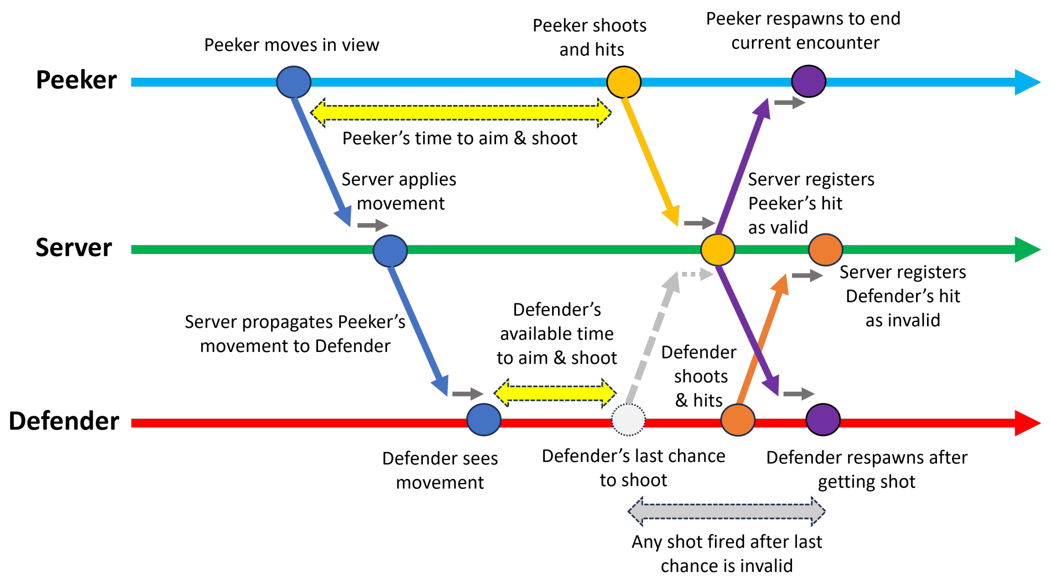

Peeker
Server
Defender
Peeker moves in view
Peeker shoots and hits
Peeker respawns to end current encounter
Peeker's time to aim & shoot
Server applies movement
Server registers Peeker's hit as valid
Server registers Defender's hit as invalid
Server propagates Peeker's movement to Defender
Defender's available time to aim & shoot
Defender shoots & hits
Defender sees movement
Defender's last chance to shoot
Defender respawns after getting shot
Any shot fired after last chance is invalid

Figure 3: Timeline illustrating peeker's advantage.

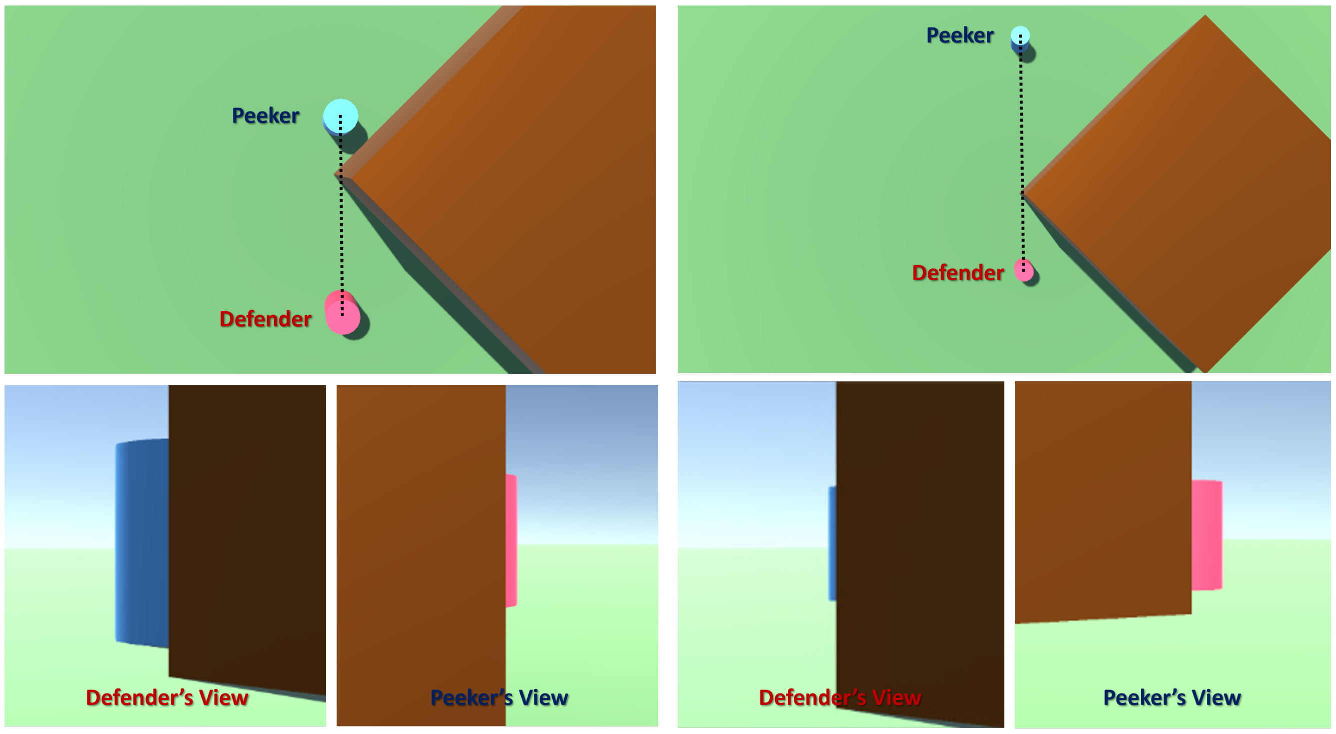


Figure 4: Avatar visibility based on distance to corner.

After this initial study was complete, we conducted a second user study as a tournament wherein the same users competed using the same game and settings, but advanced only via a single elimination bracket. Players in the bracket were unseeded, with the loser of a match being eliminated and the winner advancing. Tournament players competed for a prize (a high-end graphics card), thus providing a second data set for comparing competitive esports-like gameplay with more casual gameplay (i.e., the first, traditional user study).

Analysis of the results shows confirmation of the peeker's advantage – that latency gives the peeker a better chance of winning – but this advantage depends upon where the latency is and how large it is. A peeker's advantage increases when the peeker is further from the corner and with defender latency but not with the peeker's latency. In a more competitive environment (e.g., our tournament setting), the performance difference between peekers and defenders is diminished.

The rest of this paper is organized as follows: Section 2 provides related work, Section 3 details our user study methodology, Section 4 analyzes the results, Section 5 discusses the implications of the research, Section 6 mentions the limitations and possible future work, and Section 7 summarizes our conclusions.

2 RELATED WORK

This section summarizes studies related to our work in three main areas: latency and games, latency compensation, and player actions with latency.

2.1 Latency and Games

Numerous studies have detailed the effects of network latency and games [2, 3, 8–10, 22, 26]. Most of these studies utilize games with controlled amounts of network latency in a laboratory environment, similar to our approach in this paper, rather than observing players during normal game play. Based on this body of work, Claypool and Claypool [6] suggest game action sensitivity to latency can be classified by precision and deadline – higher precision and tighter deadlines mean more sensitivity to latency.

For FPS games, Amin et al. [2] show player experience defines and determines the sensitivity to latency for *Call of Duty*, with competitive gamers more adept at compensating for impaired conditions. Armitage et al. [3] estimate the latency tolerance threshold for *Quake 3* to be about 150-180 ms. Quax et al. [23] show that for players of *Unreal Tournament 2003* latency and latency jitter under 100 ms can degrade performance and quality of experience (QoE). Liu et al. [17] measure player performance and QoE in *Counter-Strike: Global Offensive*, finding accuracy decreases by 3% and score decreases by around 17% as latency increases to 150 ms. QoE follows suit, degrading 25% between 25 ms and 150 ms. In another study, authors demonstrate latency affects players with different skill levels similarly [15]. Their additional work shows without latency compensation network latencies manifest similarly to local system latencies [16] but generally, local latency has a greater impact. A decrease of 100 ms of local latency increased accuracy by 6%, score by 3 points/minute, and QoE by 1.6/5 points, whereas a decrease of 100 ms of network latency only results in accuracy increasing by 2%, score by 2 points/minute, and QoE by 0.7/5 points.

For games from other genres, Dick et al. [8] show via a survey that players generally think about 120 ms is the maximum tolerable latency for a network game, regardless of game genre, but their user study shows players find 150 ms acceptable. Pantel and Wolf [22] show latencies of about 100 ms can affect car racing games. Fritsch et al. [9] find players of the role-playing game *Everquest 2* can tolerate hundreds of milliseconds of network latency, while Hoßfeld et al. [10] find players of the casual game *Minecraft* are insensitive to network latencies of up to 1 second. Sheldon et al. [26] find some aspects of play in the real-time strategy game *Warcraft 3* are not affected by up to a second of network latency. Schmid et al. [25] studied two levels of latency (50 ms vs. 150 ms) and variation (0 ms vs. 50 ms), and observed that while high latency notably diminishes player performance and experience, variations of up to 50 ms do not significantly impact player performance.

While these works have been instrumental in better understanding the effects of latency on players in online games, and FPS games specifically, they have not dealt with lower-level encounters (e.g., such as a peeker attacking a defender holding a corner in an FPS game). As such, they do not illuminate the effects of latency on actual in-game encounters nor do they extrapolate to conditions outside of games. Moreover, all work to date is from general gameplay or test-based user studies and does not necessarily reflect player performance in tournaments, nor do the results compare regular gameplay with more competitive tournament gameplay.

2.2 Latency Compensation

Although our work does not address latency compensation explicitly, techniques that mitigate the effects of latency are commonly implemented in FPS games and may impact the magnitude of a peeker's advantage so are relevant here.

There are a variety of techniques that seek to mitigate the effects of network latency through software on the client or server (or both) [4, 18]. Feedback, time manipulation and prediction techniques are most commonly implemented in FPS games. Feedback approaches provide the player with information that either hides or reveals latency. Prediction techniques estimate game-state for a player's avatar based on previous known game-state. Time manipulation alters the in-game time at the server to match that of the client when resolving player actions. The effects of latency compensation on a peeker's advantage has been given little attention, but Lee and Chang [12] find some latency compensation techniques feel like "unpeeking" in that a player's avatar moves back behind a corner. Despite this, network latencies of up to 250 ms did not affect the perceived fairness of the game.

Our methodology uses latency compensation in our game client – prediction, specifically self-prediction – so as to represent a common FPS configuration playing on a PC but without any other time manipulation on the server.

2.3 Player Actions with Latency

The scenario studied in our paper – a peeking attacker encountering a stationary defender holding a corner – combines two fundamental actions common to all FPS games: movement (navigation) and shooting (target selection).

Navigation in games is when a player moves the position of an avatar from one area in a virtual world to another. Often, a player wants to navigate as quickly as possible without colliding with obstacles in the game world.

For related work in computer games, Pantel and Wolf [22] measure the time to steer a car around a race track when input is delayed by added latency. Their experiments suggest 50 milliseconds of latency is acceptable for users, but that the time to complete a lap of the track increases sharply with latency – roughly doubling for 250 milliseconds of latency.

Sabet et al. [24] investigate player QoE for navigating a car around a track for the commercial game *Need for Speed Shift 2 - Unleashed* as it relates to performance with latency. They find latency degrades player performance which, in turn, lowers the quality of experience.

Liu and al. [13] study first-person navigation, where players position their avatars to see their opponent or avoid being seen. They find network latency and local latency show similar effects in that increases in latency decrease player scores, more so for the peeker but less so for the defender.

Xu et al. [29] analyze how much time a player moves their avatar per minute in a competitive FPS game. They find that as latency increases, the player's avatar movement decreases but this depends on the weapon being used. This is relevant to our work where added latency may impact peeker movements.

Aiming in FPS games is similar to target selection, when a player moves a pointing device to an object on the screen and then "clicks" on it (e.g., by pressing a mouse button). While FPS aiming rotates the view of the world keeping the reticle in the screen, prior work has shown such selection behaves similarly to 2d selection/aiming [21]. In most cases, selecting a target as fast as possible is desirable, but accuracy is often considered, too.

Long and Gutwin [19] study the effects of latency on selecting a moving target. They find target speed directly affects the impact of latency, with selecting fast targets affected by latency as low as 50 ms, but slower targets resilient to latency as high as 150 ms. Their follow-on study [20] measures selection times for different-sized moving targets. They find that the effects of latency on selection are exacerbated by fast target speeds. Claypool et al. [7] investigate selecting a moving target with a mouse in the presence of latency. Their analysis shows selection times are impacted exponentially by latency and target speed for constant-velocity targets.

Liu et al. [14] evaluate the impact of latency on 3D target selection via a bespoke FPS environment, finding latency degrades player performance as measured by the time to select/shoot a target. They find that individual latency compensation techniques cannot fully overcome the negative effects latency has on player performance, but latency compensation techniques used in combination can lift player performance close to that of no network latency.

3 METHODOLOGY

To investigate how network latency affects the peeker's advantage in first-person shooter (FPS) games, we conducted a user study with a custom FPS game and map, controlled player roles (peeker or defender), distance from corner and latency values, and measured player performance. We then repeated the user study in a tournament setting with a prize for the winner in order to compare how player performance in a traditional study – with little at stake – differs from player performance in a competitive study where players have more to gain by winning.

3.1 Game Description

We extended an open-source FPS game – FirstPersonScience (FPSci)[1] [5] – to support multi-player networking with an authoritative server and self-prediction [18] on the clients, as in Figure 1.

[1]https://github.com/NVlabs/FPSci

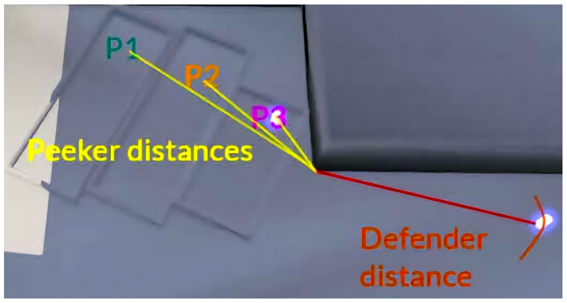


Figure 5: Top-down view of the map and player positions.

We designed a custom map that supports two players in a typical peeker-defender interaction: the defending player is stationary and faces the corner which the peeking player approaches and rounds for a firefight. Once either player is shot, both respawn to their starting locations (the defender with slight position randomness). The gameplay consists of 15 second rounds where players try to shoot each other as many times as possible. The player that scores more hits in each 15 second-round wins that round. At the end of the session, the player that wins more rounds, wins the match.

The overview in Figure 5 presents a top-down perspective of the map, indicating the starting locations of the peekers and the location of the defender. The defender is always 25 meters from the corner with the respawn position slightly randomized so as not be predictable each encounter. The defender's positional movement is disabled, but they can look around (aim) freely. The peeker has a movement bound from the corner that is restricted to a fixed distance so as to be able to get into position to see and shoot the defender but not approach closer. The peeker's movement speed is fixed at a rate similar to that of avatars in *Counter-strike: Global Offensive* – a popular commercial FPS game – and jumping is disabled. The peeker can counter-strafe – a technique competitive players use when peeking to make their avatars harder to hit.

The weapon used by both peeker and defender is single-shot (eliminates the opponent in one hit) with unlimited ammo and 500 ms delay between shots (i.e., a fire rate of 2 shots per second). These characteristics are chosen to avoid allowing players to blindly "spam" firing without aiming.

For the tournament, both players were informed that the client's gameplay screens were live-streamed over Twitch.tv both as a method of recording both players' perspectives for later analysis[2] as well as to add the atmosphere of competing in front of an audience.

We wrote a test harness that allowed the server to control all the independent variables for each round: role (peeker vs. defender), latency for each client-server connection, and peeker distance from the corner. The server and both clients logged all player actions and game states for post-study analysis.

3.2 User Study

The client computers were both gaming PCs running Windows 10 with Intel Core i7 8700 CPUs, NVIDIA GTX 1080 GPUs and 64 GB of RAM displaying on a 240 Hz Lenovo Legion Y25-25 monitor

[2]Currently, used only informally to confirm some data but could be further explored.

Table 1: Round configurations.

Round Number	Peeker's Latency	Defender's Latency	Peeker's Distance	Client Role
1	0	0	0	Peeker
2	30	0	0	Peeker
3	60	0	0	Peeker
4	0	30	0	Peeker
.	.	.	.	.
54	60	60	27	Defender

with a resolution of 1920x1080, and a right-handed Logitech G502 gaming mouse right-handed) set to 800 DPI with pointer acceleration disabled. Our methodology allowed clients to adjust the mouse sensitivity settings to suit their preferences during practice sessions prior to the matches starting. The clients and server were both on a private LAN with a GB/s Ethernet.

The local latency of our game running on our client PCs was measured with a 1000 frame/s camera (a Casio EX-ZR100) setup to capture the moment a user presses the mouse button and the resulting screen output. By manually examining the video frames, the frame time when the mouse is clicked is subtracted from the frame time the result is visible, giving the local system latency. 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.

Network latency was added to the system by delaying outgoing network packets in the client by a fixed amount before sending. In our setup, any added latency was applied to all outgoing network interfaces. For example, with 30 ms delay, a packet sent to the server and echoed back to the same client would be delayed 30 ms out and 30 ms back for a total added delay of 60 ms round-trip.

The independent variable values in our experiment are:

- Roles: Peeker, Defender
- Network latency (one-way): 0 ms, 30 ms, 60 ms
- Peeker distance from corner: 7 m, 19 m, 27 m

The distances and latencies used are typical in online FPS games and were confirmed through pilot studies.

All 27 combinations of latency and distance were pre-generated for both a peeker and defender, so that both players had a round with each setting and role exactly once, yielding 54 rounds total. These 54 different rounds are illustrated in order in Table 1, but the round order was randomly shuffled before the start of the match.

Before starting each session, the user's reaction time was collected using a custom JavaScript program that records how fast a user clicks the mouse in response to a on screen color change. The program collects 10 trials, which are averaged then reduced by the local system latency (26.2 ms) to get their average reaction time.

In summary, the study procedure (approved by our university's institute review board) for each user is as follows:

(1) Recruiting: A screener was sent to potential participants to assess their experience and skill in FPS games. Questions included most played FPS games, number of hours and skill-ranking in each game, preferred mouse sensitivity, and interest in participating in a tournament.
(2) Scheduling: Invited participants signed up for pairs of open time slots that fit their schedules.
(3) Informed consent: Before starting, each participant was given a consent form that informed them about the content of the game, video recording, and COVID protocols.
(4) Reaction time test: Each participant did a reaction time test.
(5) Practice rounds: Participants were given 18 practice rounds to get comfortable with the setup and familiarize themselves with the gameplay. Practice rounds had no added latency and a 19 m peeker distance. Participants were encouraged to adjust their mouse sensitivity between practice rounds to suit their preferences via an in-game menu/slider.
(6) Experiment rounds: The video recordings were started. The game was started on both clients and the session began once both players indicated they were ready. Play then commenced through the 54 rounds of play.
(7) Collection: After the session ended, the winner was announced based on which player had won more rounds. Then, the data files were archived and the hardware sanitized for the next session.

Each session lasted about 30 minutes. Participants received a $10 USD Amazon eGift card as remuneration for participating.

3.3 Tournament

Two months after the initial user study, 15 users that participated and indicated interest were invited to compete in a single-elimination tournament playing the same game. The tournament winner received an NVIDIA RTX 3090 graphics card as a prize. Users did not receive remuneration for participating in the tournament – the main incentive was the potential graphics card prize.

Tournament players were unseeded, so starting location in the bracket was determined randomly. We also live-streamed the tournament (participants were informed of this), advertising matches on our local media channels.

The procedure for each tournament match was the same as in the user study, as were the in-match settings, and users competed on the same computers. The tournament environment (single-elimination, prize for winning, and public spectating via Twitch.tv) is thus the only difference between the two conditions allowing for a comparison of player performance for both environments.

4 ANALYSIS

4.1 Demographics

Twenty-four (24) users participated in the study, all of whom were experienced, competitive FPS game players on a PC using a keyboard and a mouse. Users self-reported 200 to 8000 hours of FPS gameplay. Ages ranged from 18-22 years old. The average reaction time was 200 ms, faster than an average person but typical for experienced gamers [11]. Gender information was not collected, but the sample generally conformed to the predominantly male demographic of esports players (about 5% are women) and reflects the gender breakdown of FPS game players, specifically [27].

Table 2: Win percentage difference for peeker and defender for all latency combinations. Negative values (red) means the peeker is at a disadvantage and positive values (green) means the peeker is at an advantage.

		Defender		
		0 ms	30 ms	60 ms
Peeker	0 ms	-5.4% ± 2.7%	3.4% ± 2.8%	30.8% ± 2.7%
	30 ms	-7.4% ± 2.9%	4.4% ± 2.8%	28.8% ± 2.6%
	60 ms	-13.0% ± 2.8%	16.5% ± 2.8%	27.9% ± 2.7%

4.2 Results

Our analysis focuses on single interactions between a peeker and defender until one of them is shot, which we call an *encounter* – so, a single 15-second round can have multiple encounters, not strictly limited by the design, and a match has 54 rounds.

Table 2 shows the delta (difference) in average encounter win percentage calculated as the peeker's win percentage minus the defender's win percentage. For example, a -5.4% means the peeker wins the encounter about 5% less often than the defender (i.e., 47.5% peeker win, 52.5% defender win). The ± after each is a 95% confidence interval bound. Red shading indicates the defender has an advantage and green shading indicates the peeker has an advantage. From the table, with no latency for either player (top left corner), the defender has a slight advantage. This advantage is likely due to aiming by the peeker that may take a bit longer since the defender's precise location is unknown and aiming while moving is harder than aiming while stationary. Higher peeker latencies reduce any peeker advantage, whereas higher defender latencies do the opposite. In fact, for defender latencies of 60 ms, the peeker advantage is pronounced, nearly 30% regardless of the peeker's latency.

Figure 6 shows the difference in the winning percentage between the peeker and the defender based on the peeker's distance from the corner. Each point is the difference in the means over all encounters at that distance, shown with a 95% confidence interval as an error bar. From the graph, as the peeker moves farther from the corner their advantage increases, likely because the distance allows the peeker to see the defender's avatar earlier.

A one-way within-subjects ANOVA conducted to compare the effect of distance on encounter win delta for conditions of 7, 19 and 27 meters shows a significant effect; F(2,11176) = 40.7, $p < .001$.

Figure 7 shows analysis of the mean time to damage, averaged over all users and all encounters for that configuration/player. The mean time to damage is the average time it takes once a player can see (i.e., aim at) their opponent until they shoot at and hit their opponent. Visibility is calculated by lines drawn from the middle of the player avatar to the sides of the opponent avatar; if both lines intersect the corner, then the opponent is not visible. Figure 7a shows the overall mean time to damage for each role. The defender takes slightly longer (about 10 ms, or 3%) than the peeker to damage their opponent, likely because the peeker is in control of when the encounter is happening whereas the defender must anticipate and react. An unpaired t-test shows a significant difference in the overall time to damage for the peeker (M = 290.5, SD = 265.7) and defender (M = 301.0, SD = 247.5) conditions; t(12258)

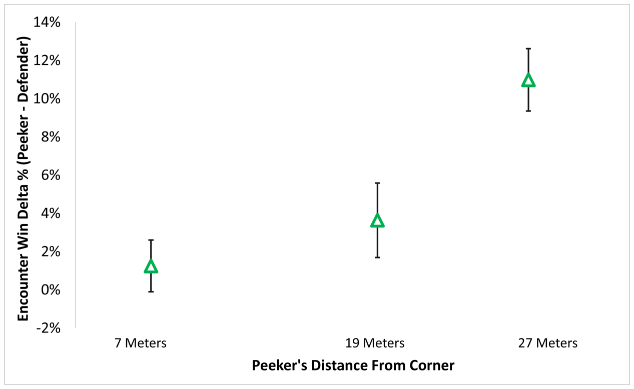


Figure 6: Win percentage difference for peeker and defender versus peeker distance from corner.

= 2.24, p = 0.03 at α = 0.05. The effect size was very small, with a Cohen's d of 0.04.

Figure 7b does the same analysis, but clusters data by peeker distance from corner, shown with a mean and 95% confidence interval.

At each distance, the peeker's time to damage is lower but somewhat surprisingly there is not a noticeable correlation in the mean time to damage and the distance. This may be due to the high skills of our participants who may be able to aim and shoot at a small target (far away) nearly as quickly as at a large target (close by).

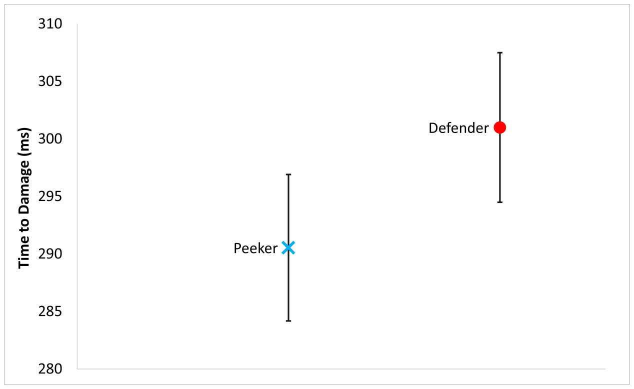
There was not a significant difference in time to damage with peeker's distance from the corner at 7 meters, for the peeker (M = 294.2, SD = 256.1) and defender (M = 304.9, SD = 213.8) conditions; t(5221) = 1.6, p = 0.10 at α = 0.05. Nor at 27 meters; peeker (M = 279.5, SD = 251.8) and defender (M = 289.8, SD = 309.9); t(3430) = 1.10, p = 0.27 at α = 0.05. But there was a significant difference at 19 meters; peeker (M = 256.6, SD = 223.2) and defender (M = 274.9, SD = 189.5); t(2472) = 2.17, p = 0.03 at α = 0.05. The effect size was very small, with a Cohen's d of 0.09.

4.3 Tournament and User Study Data Comparison

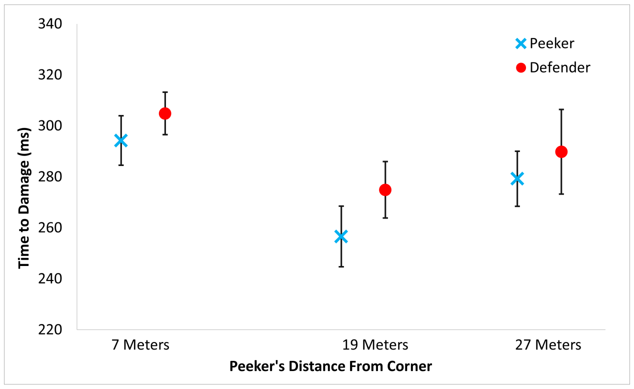
As noted, our methodology repeats our user study procedure with the only difference being that the participants are competing in a single-elimination tournament. This allows for a comparison of play in these two environments – a no stakes outcome (players get remunerated no matter if they win or lose) and a loser is out and the winner advances towards a potential attractive prize outcome (a high-end graphics card).

The tournament bracket is shown in Figure 8, where fifteen players were placed randomly (i.e, unseeded) and competed for the prize. The green boxes represent the winning player, illustrated advancing in the bracket by the bold lines.

Figure 9a shows the mean number of encounters per round for the user study and the tournament, each mean with a 95% confidence interval. The tournament users play faster, having about 20% more encounters in the same amount of time (i.e., the user study encounter rate is about 1 encounter every 2 seconds whereas the tournament encounter rate is about 1 every 1.5 seconds). There was a significant difference in average encounters per round for the



(a) Overall.



(b) Versus peeker's distance from corner.

Figure 7: Mean time to damage for peeker and defender.

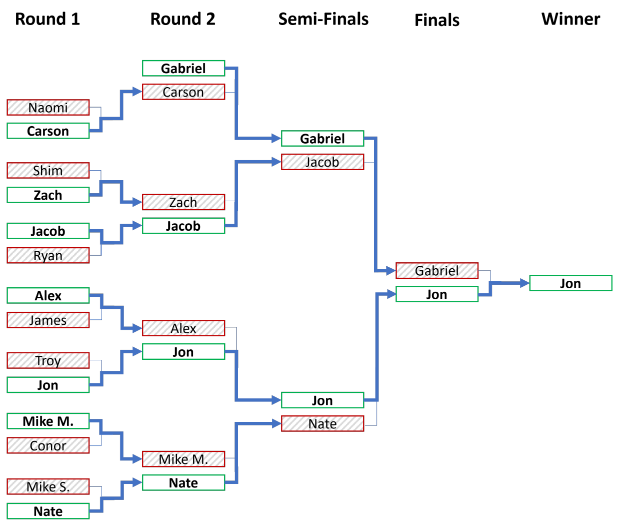
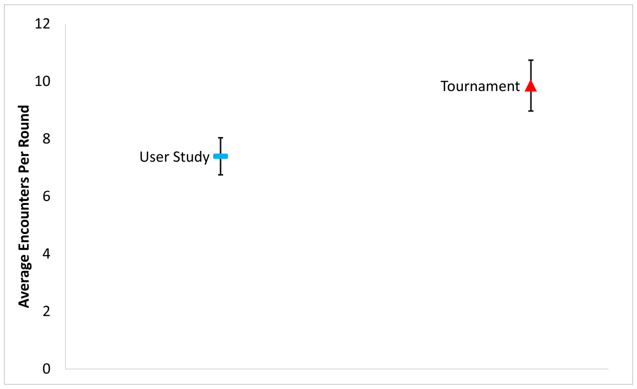
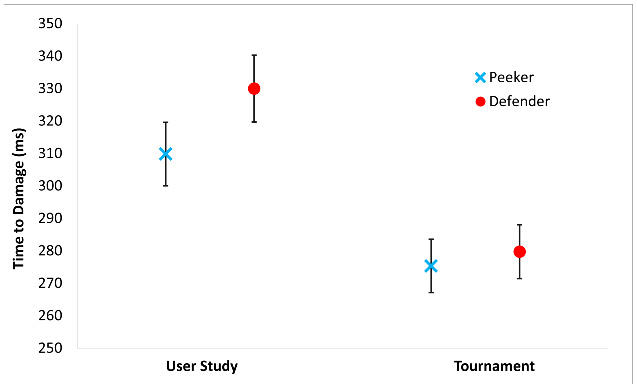


Figure 8: The tournament bracket.

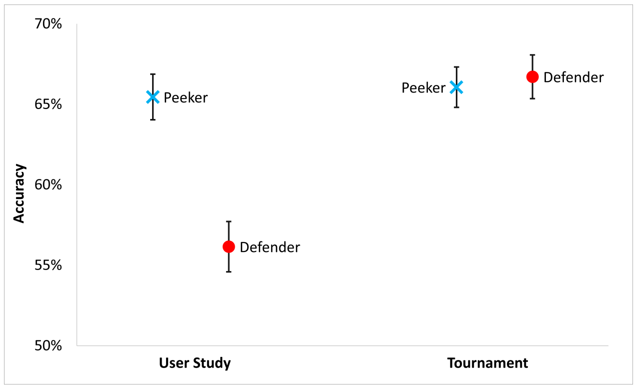
tournament (M = 9.85, SD = 3.31) and user study (M = 7.4, SD = 2.42) conditions; t(106) = 4.40, $p < .001$ at $\alpha = 0.05$ with a Bonferroni correction. The effect size was large, with a Cohen's d of 0.85.



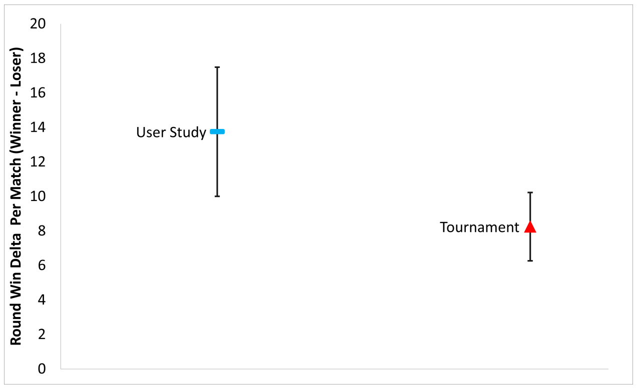
(a) Average encounters per round.



(b) Average time to damage for peeker and defender.



(c) Average accuracy for peeker and defender.



(d) Round win delta per match.

Figure 9: User study and tournament comparisons.

This trend holds for time to damage, shown in Figure 9b for the mean time to damage and 95% confidence intervals. The tournament time to damage is about 40 ms lower than in the user study. Moreover, while the peeker still has a slightly lower average time to damage, the difference between time to damage means is smaller in the tournament setting than for the user study, suggesting more competitive encounters. There was a significant difference in the user study peeker accuracy (M = 309.8, SD = 271.7) and tournament peeker accuracy (M = 275.3, SD = 255.4) conditions; t(6683) = 5.33, $p < .001$ at $\alpha = 0.05$ with a Bonferroni correction. The effect size was small, with a Cohen's d of 0.13.

There was a significant difference in the defender's time to damage in the user study defender accuracy (M = 330.0, SD = 255.5) and tournament defender accuracy (M = 279.7, SD = 239.5) conditions; t(5561) = 7.52, $p < .001$ at $\alpha = 0.05$ and a Bonferroni correction. The effect size was small, with a Cohen's d of 0.20.

This competitiveness trend is also evident in accuracy, shown in Figure 9c with means and 95% confidence intervals, where the difference in mean accuracy is about 10% for the user study but less than 2% (and not stistically significantly different) for the tournament. There was not a significant difference in peeker's accuracy for the user study peeker accuracy (M = 0.65, SD = 0.44) and tournament peeker accuracy (M=0.66, SD=0.44) conditions; t(8449) = 0.63, p = 0.53 at $\alpha = 0.05$.

There was a significant difference in defender's accuracy for the user study defender accuracy (M = 0.56, SD = 0.46) and tournament defender accuracy (M = 0.67, SD = 0.44) conditions; t(7384) = 10.01, $p < .001$ at $\alpha = 0.05$ with a Bonferroni correction. The effect size was small, with a Cohen's d of 0.25.

Figure 9d depicts an overall view of the matches, showing the mean differences in rounds won (out of 54) for the match winner versus the loser, with 95% confidence intervals. Whereas the user study had the match winner defeat the loser by an average of about 12 rounds, this margin decreased in the tournament, to only 8 rounds. There was a significant difference in round win delta for the user study (M = 13.8, SD = 9.4) and tournament (M = 8.25, SD = 5.0) conditions; t(46) = 2.55 $p = 0.014$ at $\alpha = 0.05$ with a Bonferroni correction. The effect size was medium, with a Cohen's d of 0.74.

5 DISCUSSION

In general, the data from both the user study and the tournament confirms the "common knowledge" of the peeker's advantage. Notably, higher latencies for the defender provide an advantage for the peeker, presumably because the defender has less time to react (aim and shoot) than does the peeker as the peeker comes around the corner (see Figure 3). This advantage is clear for each peeker latency (each row in Table 2) in that the peeker wins more often as the defender latency increases. However, the peeker's advantage for a given defender's latency is not consistent, with the advantage decreasing with peeker latency for a defender with no latency, increasing for a defender with modest latency (30 ms) and staying relatively constant for a defender with high latency (60 ms). There may be a "tipping point" where even modest amounts of latency impact both positions (presumably, somewhere around 30 - 60 ms), but more study is required to ascertain this. It is possible that the differences could be related to our deliberate setup where the defender is stationary (i.e., the defender avatar cannot move), which may put the defending players into a more passive mental state while the peeker is more active than is typical in FPS games where both players can move their avatars.

The results confirm that distance from the corner makes a difference in the chance of winning an encounter, most likely due to the difference in target sizes and sight lines (often called "holding angles" by competitive FPS players) for the two avatars (see Figure 4).

The tournament data illustrates that differences likely exist between casual FPS matches, where there is little riding on the outcome, and more serious matches where the outcome has consequences – in our case, elimination from the tournament. Most commercial FPS games offer both – game modes where players can practice their skills or simply play and have fun without affecting their status, and competitive play where the outcome changes their game ranking. Given there are apparent differences in the performance of each, user studies of FPS games – which typically do not have a competitive aspect in that participants get the same "benefits" no matter their performance – should take this into consideration.

6 LIMITATIONS

While our encounters featuring a peeker taking a corner held by a defender is a common, fundamental occurrence in FPS games, in our scenario the defender cannot move, unlike in most games. The restriction on movement for the defender was deliberate so as to force players into roles (peeker or defender) whereas in encounters in typical FPS game play the roles may be fluid with the defender and peeker trading roles once an encounter begins.

In order to emphasize controlled aiming and shooting, our weapon is hit-scanned (rather than firing a propagated projectile) and with a relatively low rate of fire. Results for projectile weapons (e.g., a rocket launcher) or rapid fire weapons (e.g., a machine gun) may be different.

The FPS experimentation platform we used in our study – First Person Science (FPSci) – has limited graphics, favoring fast frame rates and low latency over graphical quality. While this makes it useful for user studies with latency, such as our work, higher quality graphics including various lighting and particle effects may change encounter performance. In particular, direct target model intersections for hit detection used in FPSci – a simple cylinder shaped avatar and hitbox in our case – are less complex than most commercial FPS games where avatar hit boxes consist of several smaller hit regions and hitting different regions has a different effect (e.g., a shot to the head being different than a shot to a limb). Related to this is the audio feedback in our experiments being limited, providing a weapon fire sound effect but with no clear differentiation for a hit sound effect and, more importantly, no sound effects for footsteps where a keen-eared defender might otherwise hear an approaching peeker.

7 CONCLUSION

The growth and popularity of online games and esports – the FPS genre specifically – provides an opportunity to study skilled players to better understand how they interact with computers and perform under different system conditions.

The effects of latency on player performance has been well-studied, but typically has been done for the whole game or done by isolating a specific game action, such as aiming and shooting or navigation. Missing is detailed analysis of player-versus-player encounters and a comparison of performance at different levels of competitive intensity.

Our paper studies the *peeker's advantage* where the peeking player attacks the defending player around a corner – an encounter between two players fundamental to most FPS games. The advantage the peeker gets has been presumed to be due to latency, but the confirmation and extent of this advantage has not been studied.

To begin to address this, we conduct a user study with a custom FPS game mode and map so as to isolate players into either the peeker or defender role and control latency and peeker distance from the corner. A follow-on study with a subset of users from the original study in a tournament allows for comparison of a competitive match versus a casual environment.

Analysis of data from 24 users shows the peeker's advantage is primarily affected by the defender's latency and less affected by the peeker's latency. The peeker's advantage is the greatest when the peeker is furthest from the corner. When the same participants play in a tournament, there is more "action" each round and the matches are significantly more competitive.

Future work includes exploring some of the limitations mentioned in Section 6, including alternate weapons, moving defenders and incorporating better graphics and sound effects. Determining more precisely when and why latency differences matter – a "tipping point" – may also be useful future work to understand latency's impact.

REFERENCES

[1] Abhimannu Das. 2021. What is Peeker's Advantage in Valorant? https://tinyurl.com/peekerValorant. Accessed: November 22, 2023.

[2] Rahul Amin, France Jackson, Juan E. Gilbert, Jim Martin, and Terry Shaw. 2013. Assessing the Impact of Latency and Jitter on the Perceived Quality of Call of Duty Modern Warfare 2. In *Proceedings of HCI – Users and Contexts of Use*. Las Vegas, NV, USA, 97–106.

[3] Grenville Armitage. 2003. An Experimental Estimation of Latency Sensitivity in Multiplayer Quake 3. In *Proceedings of the 11th IEEE International Conference on Networks (ICON)*. Sydney, Australia.

[4] Yahn W. Bernier. 2001. Latency Compensating Methods in Client/Server In-game Protocol Design and Optimization. In *Proceedings of the Game Developers Conference (GDC)*. San Francisco, CA, USA.

[5] Ben Boudaoud, Josef Spjut, and Joohwan Kim. 2022. FirstPersonScience: An Open Source Tool for Studying FPS Esports Aiming. In *ACM SIGGRAPH Talks*. 2.

[6] Mark Claypool and Kajal Claypool. 2006. Latency and Player Actions in Online Games. *Commun. ACM* 49, 11 (Nov. 2006).

[7] Mark Claypool, Ragnhild Eg, and Kjetil Raaen. 2017. Modeling User Performance for Moving Target Selection with a Delayed Mouse. In *Proceedings of the 23rd International Conference on MultiMedia Modeling (MMM)*. Reykjavik, Iceland.

[8] Matthias Dick, Oliver Wellnitz, and Lars Wolf. 2005. Analysis of Factors Affecting Players' Performance and Perception in Multiplayer Games. In *Proceedings of NetGames*. Hawthorne, NY, USA.

[9] Tobias Fritsch, Hartmut Ritter, and Jochen H. Schiller. 2005. The Effect of Latency and Network Limitations on MMORPGs: a Field Study of Everquest 2. In *Proceedings of NetGames*. Hawthorne, NY, USA.

[10] O. Hohlfeld, H. Fiedler, E. Pujol, and D. Guse. 2016. Insensitivity to Network Delay: Minecraft Gaming Experience of Casual Gamers. In *Proceedings of the International Teletraffic Congress (ITC)*. Wurzburg, Germany.

[11] Chunzhen Jiang, Aritra Kundu, and Mark Claypool. 2020. Game Player Response Times versus Task Dexterity and Decision Complexity. Association for Computing Machinery, New York, NY, USA, 277–281.

[12] Steven W. K. Lee and Rocky K. C. Chang. 2017. On "Shot Around a Corner" in First-person Shooter Games. In *Proceedings of NetGames*. 1–6.

[13] Shengmei Liu and Mark Claypool. 2022. The Impact of Latency on Navigation in a First-Person Perspective Game. In *Proceedings of the ACM CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI '22)*. Association for Computing Machinery, New York, NY, USA, Article 154, 11 pages.

[14] Shengmei Liu and Mark Claypool. 2023. The Impact of Latency on Target Selection in First-Person Shooter Games. In *Proceedings of the ACM Multimedia Systems Conference (MMSys)*. Vancouver, Canada.

[15] Shengmei Liu, Mark Claypool, Atsuo Kuwahara, James Scovell, and Jamie Sherman. 2021. L33t or N00b? How Player Skill Alters the Effects of Network Latency on First Person Shooter Game Players. In *Proceedings of the Workshop on Game Systems (GameSys '21)* (Istanbul, Turkey). Association for Computing Machinery, New York, NY, USA, 6 pages.

[16] Shengmei Liu, Atsuo Kuwahara, James Scovell, Jamie Sherman, and Mark Claypool. 2021. Comparing the Effects of Network Latency versus Local Latency on Competitive First Person Shooter Game Players. In *Proceedings of the ACM Esports and High Performance HCI Workshop (EHPHCI)*. Virtual Conference.

[17] Shengmei Liu, Atsuo Kuwahara, Jamie Sherman, James Scovell, and Mark Claypool. 2021. Lower is Better? The Effects of Local Latencies on Competitive First Person Shooter Game Players. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI)*. Virtual Conference.

[18] Shengmei Liu, Xiaokun Xu, and Mark Claypool. 2022. A Survey and Taxonomy of Latency Compensation Techniques for Network Computer Games. *ACM Comput. Surv.* 54, 11s, Article 243 (Sept. 2022), 34 pages. https://doi.org/10.1145/3519023

[19] Michael Long and Carl Gutwin. 2018. Characterizing and Modeling the Effects of Local Latency on Game Performance and Experience. In *Proceedings of the Annual Symposium on Computer-Human Interaction in Play (CHI Play)*. Association for Computing Machinery, New York, NY, USA, 285–297.

[20] Michael Long and Carl Gutwin. 2019. Effects of Local Latency on Game Pointing Devices and Game Pointing Tasks. In *Proceedings of the ACM CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, Glasgow Scotland, UK, 1–12.

[21] Julian Looser, Andy Cockburn, and Joshua Savage. 2005. On the Validity of Using First-Person Shooters for Fitts' Law Studies. *People and Computers XIX* 2 (2005).

[22] Lothar Pantel and Lars C. Wolf. 2002. On the Impact of Delay on Real-Time Multiplayer Games. In *Proceedings of the Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*. Miami, FL, USA.

[23] 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 *Proceedings of NetGames*.

[24] Shafiee Sabet Saeed, Steven Schmidt, Saman Zadtootaghaj, Carsten Griwodz, and Sebastian Moller. 2018. Towards Applying Game Adaptation to Decrease the Impact of Delay on Quality of Experience. In *Proceedings of IEEE International Symposium on Multimedia (ISM)*. 114 – 121.

[25] Andreas Schmid, David Halbhuber, Thomas Fischer, Raphael Wimmer, and Niels Henze. 2023. Small Latency Variations Do Not Affect Player Performance in First-Person Shooters. In *Proceedings of the ACM on Human-Computer Interaction 7 (CHI PLAY)*. New York, NY, USA, 197–216. https://doi.org/10.1145/3611027

[26] Nathan Sheldon, Eric Girard, Seth Borg, Mark Claypool, and Emmanuel Agu. 2003. The Effect of Latency on User Performance in Warcraft III. In *Proceedings of NetGames*. Redwood City, CA, USA.

[27] Statista. 2020. Distribution of Gamers Playing Selected Game Genres Worldwide as of January 2017, by Gender. Online: https://tinyurl.com/yytrbj4d. (Accessed September 17, 2020).

[28] Verified Market Reports. 2023. Global Shooting Games Market Synopsis. https://tinyurl.com/shooterReport. Accessed: Feb 11, 2024.

[29] Xiaokun Xu, Shengmei Liu, and Mark Claypool. 2022. The Effects of Network Latency on Counter-strike: Global Offensive Players. In *14th International Conference on Quality of Multimedia Experience (QoMEX)*. 1–6.