Game Input with Delay - Moving Target Selection Parameters

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

Andy Cockburn
University of Canterbury
Christchurch, New Zealand
andy@cosc.canterbury.ac.nz

Carl Gutwin
University of Saskatchewan
Saskatoon, Saskatchewan, Canada
gutwin@usask.ca

ABSTRACT

All real-time computer games, particularly networked computer games, have a delay from when a player starts an action (e.g., clicking the mouse) until the game renders the result (e.g., firing a projectile). This delay can degrade both player performance (e.g., lower game score) and quality of experience (e.g., less fun). While previous work has studied the effects of delay on commercial games and individual game actions, a more detailed understanding is needed of the effects of delay on moving target selection with realistic target motion. This paper presents an in-depth study of the effects of delay on the fundamental game action of selecting a moving target with a mouse with parameters for the target motion – turn frequency and turn angle. We design and implement a custom game where players select moving targets using a mouse, while the game controls both the target motion and input delay. Analysis of data gathered in a 56-person user study shows both user time and accuracy in selecting the moving target degrade with delay. Target turn frequency and turn angle can make a target easier to select (i.e., selection accuracy increases), but take longer to do so (i.e., selection time increases), because turning slows targets down while making them less predictable.

CCS CONCEPTS

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

KEYWORDS

lag, user study, target selection

ACM Reference Format:
Mark Claypool, Andy Cockburn, and Carl Gutwin. 2019. Game Input with Delay - Moving Target Selection Parameters. In *10th ACM Multimedia Systems Conference (MMSys '19), June 18–21, 2019, Amherst, MA, USA.* ACM, New York, NY, USA, Article 4, 11 pages. https://doi.org/10.1145/3304109.3306232

1 INTRODUCTION

The computer games market is expected to be worth over 90 billion U.S. dollars by 2020, up from nearly 79 billion in 2017 (Pricewaterhouse Coopers, BestTheNews, 2016). There are more than 2.5 billion computer gamers all over the world (The European Mobile Game Market, 2016) who are increasingly playing games online. Online games alone are projected to be worth about 30 billion U.S. dollars (Capcom, International Development Group, 2017) and online PC games are expected to take 47% of the global PC and console gaming revenues in 2019 (DigiWorld, 2016).

These online games must deal with delay since player game input travels from the player's local computer and game world to game servers and other players, encountering processing and network delays on end-host computers and intermediate network devices. For traditional online games, network delays can manifest in inconsistent state between the clients and servers. For cloud-based games, network delays add at least a round-trip time from the client to the server in response to the player action [7]. Even non-networked games face local delays from the player action (e.g., moving the mouse) to rendering the effects in-game (e.g., cursor changing position) of up to 100 milliseconds [31]. Real-time games require players to make many time-sensitive actions that degrade when delayed and even delays as small as milliseconds can hamper the interplay between players' actions and intended results. For example, delay when aiming a virtual weapon in a shooting game can make it difficult for a player to hit a moving target, decreasing the player's score and degrading the quality of experience.

There are established methods to compensate for network delays in traditional online games [4, 17], such as system-level treatments (e.g., real-time packet priorities), delay compensation algorithms (e.g., dead reckoning, sticky targets, aim dragging) and even game designs that can mitigate perceived delay (e.g., deferred avatar response, geometric scaling). However, most of these techniques are not effective for local delays nor can all of them be used for cloud-based games. A detailed understanding of how delay affects player actions in games is needed in order to: 1) provide a better understanding for game developers to design levels and choose target motion parameters to account for delay, 2) develop and deploy techniques suitable for mitigating delay in local systems and cloud-based games, and 3) construct models of the effects of delay on player actions useful for game designers.

Previous work has studied the effects of delay on commercial games, often clustering work based on genre – e.g., First Person Shooter [2, 3, 17, 29], Massively Multiplayer Online [6, 13], Real-time Strategy [8], and Sports [25, 26]. While helpful for providing insights into the effects of delay for traditional network games, such studies generally cannot tease out specifics on how delay affects individual game actions, particularly since game clients can sometimes act on user input immediately, resulting in inconsistent game state between client and server instead of having the input feel laggy. Moreover, commercial games often deploy latency compensation techniques which further obfuscate the true effects of delay on user actions.

Other related work has looked at the effects of local delay on user input in games, including models of user performance [22, 27, 30]. Some work on moving target selection has suggested an exponential model of performance with delay and target speed [11], but has done so only for targets with constant velocity. In many games, targets move with complicated motion parameters, using force-based physics for acceleration and velocity and abrupt target directional changes. For example, vehicles in most first-person shooter games, such as *Halo* (Microsoft Studios, 2001-2010) or *Battlefield* (Electronic Arts, 2002-2016), and players in sports games such as *Madden NFL* (EA Sports, 1988-2018) or *Strikers Charged* (Nintendo, 2016), use force-based physics for movement.

We design and implement a game that isolates the fundamental action of selecting a realistically moving target with a mouse, controlling: a) the delay between the user input and the rendered action, and b) the target motion parameters – specifically, turn frequency and turn angle. The game records the elapsed time when the user first clicks on the target and distance from the mouse cursor when clicked and the target, and also gathers quality of experience (QoE) ratings on the perception of delay by the user. We deploy our custom game in a user study with 56 participants, using delays of 50 to 300 milliseconds, and target turn intervals of 0 to 150 milliseconds and turn angles from 0 to 360 degrees.

Analysis of the results shows user accuracy (measured by the distance from the cursor to a target when the mouse is clicked) and user selection time (measured by the elapsed time to select the target) degrade linearly over the range of delays tested – this is in contrast to earlier work [11] that showed an exponential relationship, albeit over a wider range of delays. The user is least accurate when the target turns with a mid-range (90 degree) angle, since sharper turns tend to slow the target when using realistic movements. In contrast, higher turn frequency actually makes targets easier to hit since more frequent turning decreases average target speed as targets may turn back towards the user selector (mouse), but users take longer to select with higher turn frequencies since target movements are less unpredictable. User opinions on the quality of experience (perception of delay) show a pronounced degradation even over the modest delay range tested.

The rest of this paper is organized as follows: Section 2 describes work on user input modeling, network games, and game actions related to our work; Section 3 describes our methodology, including developing our game and conducting a user study; Section 4 analyzes the results from the user study; and Section 5 summarizes our conclusions and presents possible future work.

2 RELATED WORK

This section presents related research in user input for target selection and delay: models of user input (Section 2.1), impact of delay on network games (Section 2.2) and impact of delay on individual game actions (Section 2.4).

2.1 Models of User Input

Paul Fitts pioneered early seminal work in the area of human-computer interaction and ergonomics in the form of creating *Fitts' Law* [12]. Fitts' Law describes the time to select a stationary target based on the target distance and target width. Fitts' Law has been shown to be applicable to a variety of conditions (e.g., underwater [21]) and input devices (e.g., eye tracking [35]), and has been extended to two dimensions [23], suitable for, say, target selection with a computer mouse [34], but by itself accounts for neither moving targets nor delay.

Jagacinski et al. [18], Hajri et al. [14], and Hoffmann [15] extended Fitts' Law to moving targets, enhancing the model with a target speed. However, model fit and validation was done with very few users.

Hoffman [16] and MacKenzie and Ware [24] revised Fitts' Law to consider delay, with experiments showing users have two types of responses in the presence of delay: 1) *move-and-wait* where a user provides input, then stops and waits for it to occur, repeating as necessary (less than 700 milliseconds of delay); and 2) *continuous* where a user provides continuous input in the presence of delay, adjusting as necessary without stopping (more than 700 milliseconds of delay). Jota et al. [20] studied the impact of delay on target selection and dragging with touch devices.

In general, the prior work in user input models does not account for delay and target objects that move with realistic motion, nor have prior input models been validated with a lot of users and a wide range of conditions.

2.2 Network Games

There have been numerous studies exploring the effects of delay on traditional network games.

Quax et al. [29] studied the effect of delay on a first person shooter (FPS) game, *Unreal Tournament 2003* (Epic Games, 2002). Subjective and objective measurements showed that users are influenced by even small amounts of delay.

Fritcsh et al. [13] examined delay and player performance in a massively multiplayer online role-playing game (MMORPG), *Everquest II* (Sony, 2004). User studies with controlled amounts of delay showed EverQuest's in-game strategies for coping with delay provided smooth gameplay even at delays as high as 1250 milliseconds.

Amin et al. [2] studied the effects of delay on player experience for another FPS game, *Call of Duty Modern Warfare 2* (Activision, 2009), focusing on network congestion. Their results suggested that delay jitter as low as 100 milliseconds can lower Mean Opinion Score (MOS) below 3 (a typical acceptability threshold), with some correlation with player skill.

Chen et al. [6] analyzed network traces from another MMORPG, *Shenzhou Online* (UserJoy Technology, 2004), inferring network congestion levels that lead to players quitting the game. They confirmed a pronounced correlation between game session times and minimum delay the players experience.

Broadly, all these approaches treat an existing game as a "black box" in that individual game actions are not studied in isolation. While useful for a general understanding of the effects of delay on network games, game design and game engines may obfuscate important system details, making it unclear how the findings relate to specific player actions. Moreover, network delay in traditional online games often manifests itself in an inconsistent state on the client and the server since the client can act on player input immediately instead of having the player's input wait for a round-trip time to the server.

2.3 Cloud-based Games

There is also research, albeit less, on the effects of delay in cloud-based games. Unlike in traditional network games where a client can potentially act on user input immediately, in cloud-based games all user input is delayed by at least a round-trip time to the server.

Chen et al. [7] discussed the effects of network delay, packet loss, and bitrate on frame rates and graphics quality for the cloud game systems OnLive[1] and StreamMyGame.[2] The authors did not explicitly measure player performance with delay.

Jarschel et al. [19] conducted a user study in an emulated cloud gaming system, measuring the quality of experience for games users selected to play. Sackl et al. [32] analyzed the relationships between delay and player experience for cloud gaming, showing even small delays had have a visible impact on experience, depending upon the nature of the game.

These cloud-based research works are more closely related to our work than traditional network game research since the delays studied – player input to rendered output – closely match the manifestation of delays in our work. However, these papers did not analyze individual game actions with delay.

2.4 Game Actions

An alternative approach to using a full game in a user study is to explore the effects of delay on individual (aka *atomic*) game actions.

Claypool and Claypool [10] presented a general framework describing delay and game actions that includes *precision* – the accuracy required to complete the action successfully, and *deadline* – the time required to achieve the final outcome of the action. While helpful for explaining the effects of delay on games, the precision-deadline framework did not quantify the many possible parameters that make up game actions (e.g., dodge frequency).

Raeen and Eg [30] conducted experiments with a simple button + dial interface, letting users adjust delay based on their perceptions. They found users were capable of perceiving even low amounts of delay (around 66 milliseconds).

Long and Gutwin [22] studied the effects of delay on an "atom" of interaction in a simple game – moving an object to intercept a target. They found target speed directly affected the impact of delay, with fast targets affected by delays as low as 50 milliseconds while slower targets made the action resilient to delays as high as 150 milliseconds.

Claypool et al. [11] investigated selecting a moving target with a mouse in the presence of delay, similar to our paper. Their analysis showed target selection time is impacted exponentially by delay and target speed for constant-velocity targets.

Pavloyvych and Stuerzlinger [28] and Pavloyvych and Gutwin [27] studied target selection and following for objects moving with Lissajous curves (smooth curves, but with varying sharp turns within the curve). They found tracking errors increased quickly for delays over 110 milliseconds but the effects of target velocity on errors was close to linear.

[1]https://en.wikipedia.org/wiki/OnLive
[2]https://en.wikipedia.org/wiki/StreamMyGame

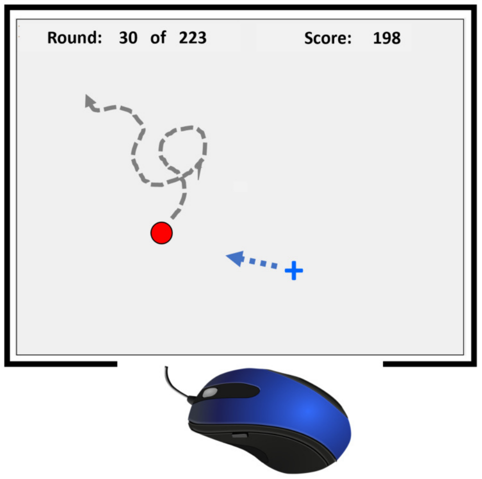


Figure 1: Juke! Users move the cursor (blue cross) with a mouse and click on a moving target (red circle). The game adds delay to the user input (both mouse movement and mouse clicking) and controls the target jink frequency and jink angle using force-based movement.

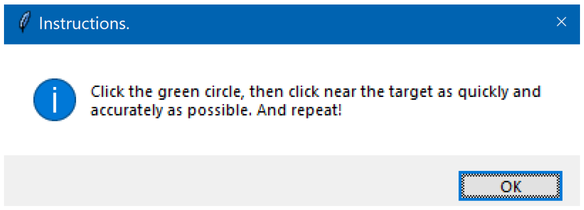
In general, while these approaches have helped understand delay and fundamental game actions, they used only basic parameters for target movements (e.g., targets with constant velocity). In many games, target objects have complicated movement paths that depend upon how often and how sharply they turn, governed by forces applied in the intended direction.

3 METHODOLOGY

To model moving target selection parameters on game input with delay, we: 1) develop a simple game (called *Juke!*) that enables study of user input with controlled delay and target motion using realistic motion parameters (Section 3.1); 2) conduct a user study to evaluate the impact of target motion parameters with delay (Section 3.2), and 3) analyze the core results of the user study (Section 4).

3.1 Juke!

In order to avoid the monotony that often occurs during behavioral experiments, we designed the target selection task as a simple game. Our custom game is called *Juke!*, depicted in Figure 1. Juke! allows study of a single user action, target selection, with controlled amounts of delay and controlled motion parameters for the target. Unlike previous studies (see Section 2.4), Juke! uses complicated target motion common to many games, where target movement is governed by force-based physics (e.g., acceleration) and targets turn (jink) sharply. In Juke!, the user proceeds through a series of short rounds, where in each round the player moves the mouse cursor (a blue '+') and attempts to select the target (a red ball) as quickly and accurately as possible. The progress to game completion through the rounds is displayed in the top left corner.


Instructions.

Click the green circle, then click near the target as quickly and accurately as possible. And repeat!

Figure 2: The Juke! instructions shown to the user.

Term	Definition
force-based physics	constant acceleration in target's direction
jink/turn	change in target direction
jink interval	time between turns
jink angle	maximum angle range for turn

Table 1: Definition of terms.

The user's score is displayed in the top right corner, where score is a running total of the distance of the cursor from the target when clicked (lower is better).

Upon starting the game, users are instructed to select the target as quickly and as accurately as possible via the dialog box shown in Figure 2.

The action chosen – selection of a 2D, moving target with a mouse – is common to many PC game genres. Some examples include: 1) top-down shooters (e.g., Figure 3 – *Nuclear Throne*, Vlambeer, 2015) where the player aims a projectile at opponents by moving the mouse to the intended target; 2) first person shooters (FPS) (e.g., Figure 4 – *Call of Duty*, Activision, 2003) where players use the mouse to pan the game world to align a reticle over a moving opponent and shoot; and 3) multiplayer online battle arenas (MOBAs) (e.g., Figure 5 – *League of Legends*, Riot Games, 2009) where players move a skill shot indicator with a mouse to target a moving opponent with a spell.

Figure 3: *Nuclear Throne* (Vlambeer, 2015), the shotgun reticle is moved with a mouse, selecting moving targets.

Juke! is written in Python using `tkinter`, and runs in full screen mode at 1080p resolution (1920x1080 pixels) and 30 f/s. For clarity, Table 1 provides the definition of terms used in describing Juke! behavior.

The user begins each round by clicking a small green circle in the middle of the screen. This situates the user's mouse cursor at the same starting location each round. Upon clicking, the green circle disappears and a red target appears at a random location a short distance from the center of the screen.

Figure 4: *Call of Duty* (Activision, 2003), the world is panned with a mouse, selecting moving opponents with the sniper reticle.

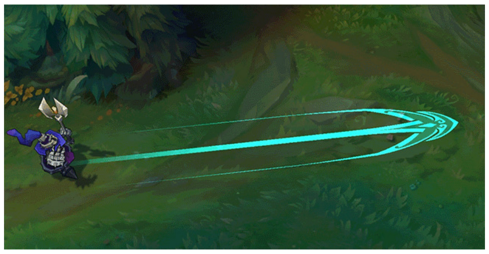


Figure 5: *League of Legends* (Riot, 2009), the skill shot's direction is moved with a mouse, selecting moving targets.

The target moves with force-based physics [5], applying an acceleration in the target's intended direction, with a limit on the maximum speed. The intended direction is adjusted based on how often the target jinks and how sharply the target changes direction when jinking. The algorithm is depicted in Figure 1. The direction of the target is initially randomly chosen. After that, each game loop (30 Hz), the direction changes every jink interval, adjusting the direction in an amount randomly chosen in the jink angle range (effectively, the target can turn at most $\frac{1}{2}$ the angle range to the left or $\frac{1}{2}$ the angle range to the right). A force of 5 pixels/s^2 is applied in the target's direction, added to the velocity. The magnitude of the velocity (maximum speed) is limited to 50 pixels/s. The velocity is added to the target's location.

The jink interval and jink angle parameters are shown in Table 2a and Table 2b, respectively. The jink interval is the time between turns. The jink angle is the maximum angle range over which the target can turn where each turn angle is chosen randomly over this range. The actual turn angle chosen can affect the speed, since turning back in the opposite direction has a braking (decelerating) effect. Effectively, these parameters provide for a range of target movements that might be experienced in shooting/combat games – e.g., fast, predictable targets (jink angle 0, jink interval 0), somewhat fast, unpredictable targets (jink angle 90, jink interval 75), slow, unpredictable targets (jink angle 360, jink interval 30).

Listing 1: Force-based Target Motion

```
// Constants.
max_speed = 50            // pixels/s
force = 5                 // pixels/s^2

// Initial values.
direction = random(360) // degrees
velocity.x, velocity.y = 0, 0

// Repeat at 30 Hz until mouse clicked.
each game loop

  // If long enough, random turn based on angle range.
  if elapsed > jink_interval then
     turn = random(1) × jink_angle // pick in range
     turn = turn - $\frac{1}{2}$ jink_angle    // left or right
     direction += turn
  end if

  // Update velocity based on force in direction.
  force.x = $\cos(direction \frac{\pi}{180}) \times force$
  force.y = $\sin(direction \frac{\pi}{180}) \times force$
  velocity.x += force.x
  velocity.y += force.y

  // Limit speed to maximum.
  speed = $\sqrt{velocity.x^2 + velocity.y^2}$
  if speed > max_speed then
    velocity.x *= $\frac{max\_speed}{speed}$
    velocity.y *= $\frac{max\_speed}{speed}$
  end if

  // Update position based on velocity.
  location.x += velocity.x
  location.y += velocity.y

  ...
end game loop
```

Milliseconds:
30, 75, 150

(a) Jink Intervals.

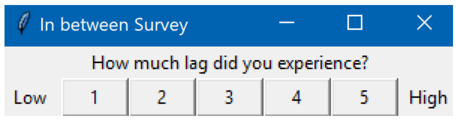
Degrees:
0, 90, 360

(b) Jink Angles.

Milliseconds:
0, 62.5, 125, 250

(c) Added Delays.

Table 2: User study parameters.



How much lag did you experience?

Low 1 2 3 4 5 High

Figure 6: Quality of experience prompt given to user.

The game adds a controlled amount of delay selected from the set in Table 2c. The delay is added to all user input (mouse movement and clicks) for the duration of the round. Each jink interval, angle & delay combination appears 5 times, but the entire set of combinations is shuffled into a different random order for each user.

Exactly once for each combination of jink interval, angle & delay, the user is asked to rate the quality of experience (QoE) based on the perceived amount of lag experienced during the previous round, shown in Figure 6. The game pauses until the user selects a choice, 1 (low) to 5 (high).

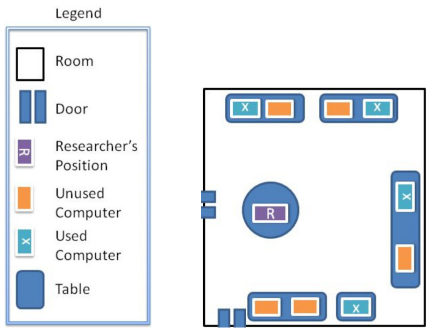


Figure 7: Lab for user study [9].

For practice, the first 3 rounds have no added delay and slow (half acceleration) targets, with a jink interval of 150 ms and a jink angle of 90 degrees. After the practice rounds, the user plays through the 36 jink rounds (4 delays x 3 intervals x 3 angles), 4 no turn rounds (4 delays), and 1 stationary target (4 delays), 5 times each, for a total of $3 + (36 + 4 + 4) \times 5 = 223$ rounds, all shuffled randomly. Between rounds, the user is free to pause as long as necessary to rest/regain concentration. Playing through the entire set of rounds typically takes less than 15 minutes.

3.2 User Study

Our user study was conducted in windowless but brightly lit computer lab (layout shown in Figure 7, as used in previous user studies [9]). The lab computers had Intel i7-4790 4 GHz processors, 16 GB of RAM, and NVidia GeForce GTX 960 graphics cards and ran Microsoft Windows 10. The PC monitors were Dell U2412M 24" LCDs displaying 1920x1200 pixels @ 59p Hz.

Study participants were solicited via University email and a system whereby Psychology students participate in user studies to obtain class and major credit.[3] Additional incentives were a raffle for a $25 gift card and for the user with the lowest total score. Participants from an author's class also received 1 extra point on an exam.

Upon arrival in the lab, users were read a script about the study and asked to sign an IRB consent form. Then users were encouraged to adjust the computer chair height and monitor angle/tilt so as to be comfortably looking at the center of the screen. Users were told to they could use the mouse with whichever hand was preferred.

Next, users completed a demographics and gaming experience survey coded using Qualtrics.[4] Lastly, the game and incentives were described and the game launched.

Users immediately started playing. However, the first three rounds had no added delay and slow targets for practice and the results were not recorded. Users then played through all 5x shuffled combinations of jink interval, angle & delay, (Table 2a, Table 2b, and Table 2c), with one QoE question (Figure 6) for each jink interval-angle-delay combination.

[3]http://wpi.sona-systems.com/
[4]https://www.qualtrics.com/

3.3 Local System (Base) Delay

Note, the delays in Table 2c added by Juke! are in addition to any delays inherent in the base computer system. Since such base delays have been shown to be significant [17, 31], we measured the base delay in Juke! on our lab computers using a Blur-busters type technique.[5]

An LED on a bread board was connected via wire to a mouse so that the LED lit when the mouse button was clicked. A high frame rate camera (Casio EX-ZR100) filmed the player at 1000 f/s, capturing the moment the QoE prompt was selected using the mouse. By manually examining the individual video frames, the frame number when the light appeared when the mouse was clicked was subtracted from the frame number when the QoE prompt showed the input, giving the base delay (in milliseconds).

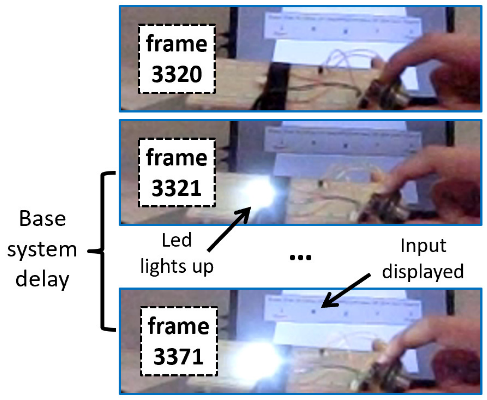


Figure 8: Measuring base delay.

Figure 8 depicts the measurement method. The cursor is poised over the QoE prompt in frame 3320. In frame 3321, the LED on the breadboard is lit since the mouse has been clicked. The resulting output (the QoE prompt disappearing) is not seen until frame 3371. Since there is one video frame each millisecond, subtracting 3371 from 3321 gives a base delay of 50 milliseconds.

The measurement method was repeated 5 times, resulting in base delay values of 50, 49, 52, 53 and 48 milliseconds. Hence, 50 milliseconds is added to all delay analysis.

4 ANALYSIS

This section first summarizes participant demographics (Section 4.1), then presents the core results – the user performance in the presence of delay (Section 4.2). Further analysis investigates target selection time by player skill (Section 4.3) and compares the results with other studies of game actions (Section 4.4) and traditional network games (Section 4.5). The section ends with brief analysis of the user Quality of Experience (QoE) (Section 4.6).

4.1 Demographics

The study had fifty-six participants, aged 17-26 years (mean and median 20). Sixteen users were female and 40 male. Fifty-two were right-handed, 3 left-handed and 1 ambidextrous. User self-rating as a PC gamer (scale 1-low to 5-high) had a mean of 3.5, with a slight skew towards having "high ability". Half of the users played 6 or more hours of computer games per week. Most users majored in Robotics Engineering, Computer or Game Development.

[5]http://www.blurbusters.com/gsync/preview2/

4.2 User Performance

There are two primary metrics for user performance: a) *selection time* – the elapsed time from the start of the round until the mouse button is clicked; and b) *accuracy* – the distance from the mouse cursor to the target when the mouse button is clicked.

Figure 9 shows graphs of cumulative distribution functions for selection time and accuracy. For the top graph (selection time), the x-axis is the elapsed time when the mouse button is clicked, and for the bottom graph (accuracy), the x-axis is the distance (in pixels) from the mouse to the target when the mouse button is clicked. For both graphs, the y-axis is the cumulative distribution. For both graphs, there is a clear separation of the trendlines with delay, with the distributions for the lower delays towards the left (lower) and the distributions for the higher delays towards the right (higher). All CDFs have a heavy tail, with the top 5% of the distribution significantly higher than the body. The minimum elapsed times are around a half-second, but can be as high as around 4 seconds.

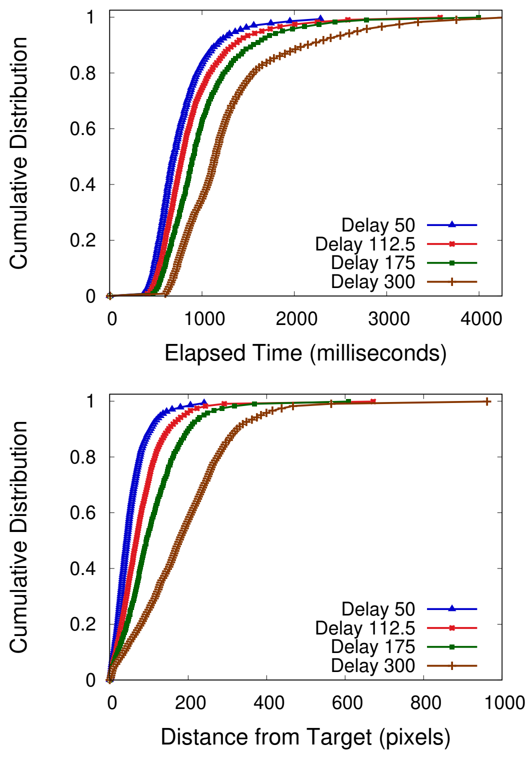


Figure 9: Cumulative distribution functions for user selection time (top) and user accuracy (bottom), clustered by delay (milliseconds).

Figure 10 depicts graphs of selection time and accuracy versus delay. For the top graph (selection time), the y-axis is the elapsed time when the mouse button is clicked, and for the bottom graph (accuracy), the y-axis is the distance (in pixels) from the mouse to the target when the mouse button is clicked. For both graphs, the x-axis is the total input delay (added delay + base delay). Each point is the mean time for all users for that delay, shown with a 95% confidence interval. Overall, there is a linear increase in both selection time and accuracy with delay over the range of delays tested. This is in contrast to previous work [11] that showed an exponential relationship with delay, albeit the ranges of delays in the previous study were larger (maximum 450 milliseconds versus 300 milliseconds).

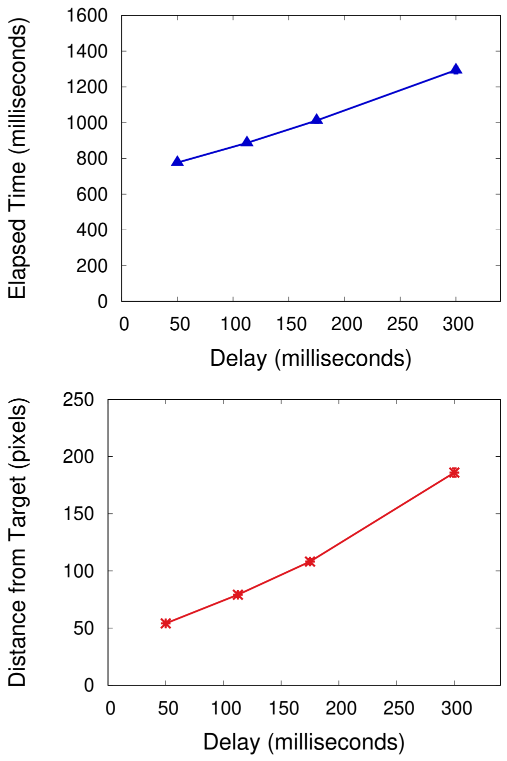


Figure 10: User selection time (top) and user accuracy (bottom) versus delay.

Figure 11 graphs the same data as in Figure 10, but with data grouped by the target jink angle (the angle range used when a target turns). Both graphs have three trendlines, one for each target jink angle. Each point is the mean distance for all users for that delay & angle combination, shown with a 95% confidence interval. Generally, there is a clear separation of the trendlines for both graphs, indicating target jink angle affects user selection time and accuracy. For selection time, the angle 360 targets are the hardest to select, indicated by the longest times, the angle 90 targets the easiest and the no jink targets in between, but closer to the angle 90 targets. Delay impacts the elapsed time about the same for all jink angles. In contrast, for accuracy, the angle 90 targets are the hardest to select, indicated by the largest distances, the angle 360 targets the easiest and the no jink targets in between. Delay impacts the distance for all jink angles, but is most pronounced for the angle 90 targets and the trend lines diverge as delay increases.

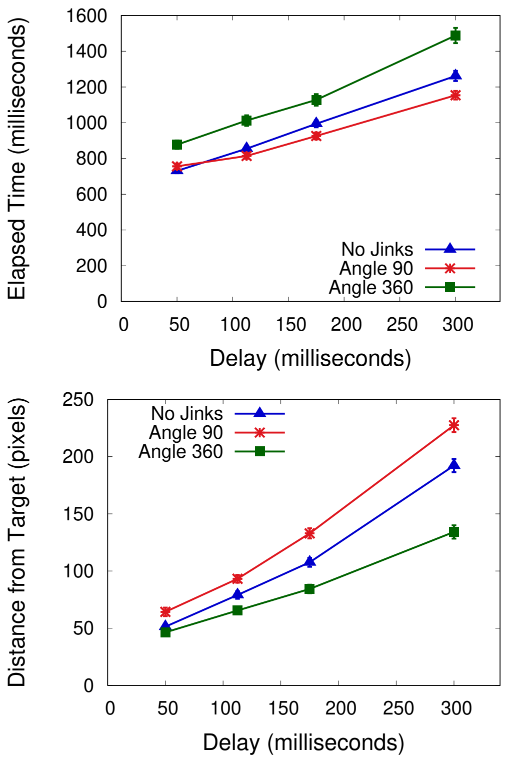


Figure 11: User selection time (top) and user accuracy (bottom) versus delay, data grouped by target jink angle.

The effects of the target jink angle are illustrated in Figure 12, which depicts the target paths for three sample targets, one for each angle tested. The point density (1 point per 30 milliseconds) on all paths indicates the relative target speeds, with points spaced further apart indicating higher target speed. When the jink angle range is large (e.g., angle range 360 degrees) the target frequently reverses direction, causing the applied force to brake the target speed, making it easier to hit. However, the large angle range makes the target path unpredictable, causing the user to take longer to track target location and click the button. On the other hand, when the target does not turn (e.g., angle range 0 degrees), while the target moves more quickly, tracking the target location is relatively easy since it moves in a straight line. When the target makes more gradual turns (e.g., angle range 90 degrees), a turn does not drastically slow the target, but it does make it harder to predict the location for tracking and selection.

We surmise there is one jink angle where users are most inaccurate when selecting and another where users are slowest to select the target. Figure 13 shows analysis that supports this idea. The y-axes for both graphs are as for Figure 10, but here the x-axis is the jink angle range. Data points are still means with 95% confidence intervals. For selection time, the largest angle range takes

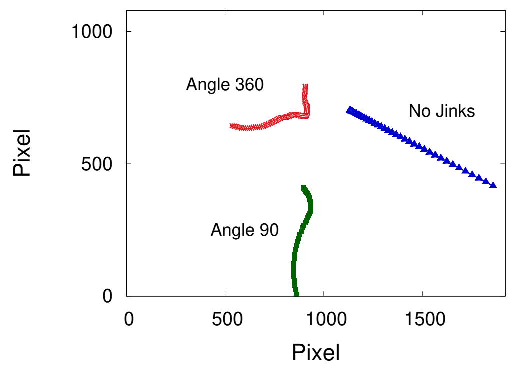


Figure 12: Example paths traveled by 3 targets – no jinks, jink angle 90, jink angle 360.

the longest to select, while the angle 0 and angle 90 angle ranges take less time to select. For accuracy, the largest angle range is actually the easiest to hit (the distance is the lowest), while the 90 degree angle range is the hardest to hit of the 3 angles tested. Future tests with more angle ranges are needed to accurately determine the shape of the curves over the full 0-360 angle range.

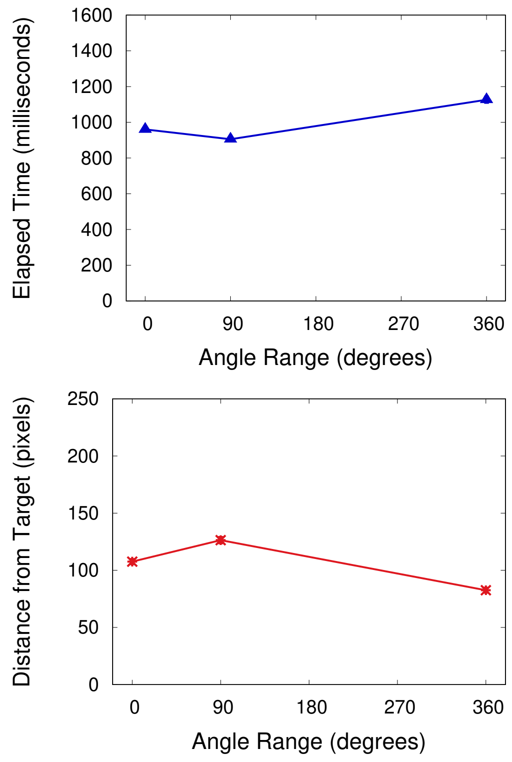


Figure 13: User selection time (top) and user accuracy (bottom) versus jink angle range.

Figure 14 graphs the same data as in Figure 10, but the data is grouped by the jink interval (the time between target turns). The non-jink case is shown for reference. For selection time, the target that turns the most often (jink interval 30 milliseconds) takes the longest to select, while the time to select targets with higher turn intervals is similar to the time to select targets that do not turn. For accuracy, targets that turn less often are harder to hit. However, not turning at all (no jinks) makes a target harder to hit than frequent turns (jink interval 30 milliseconds), likely because of the effects of frequent turns on target speed. As for jink angle, this suggests there are different jink intervals that make a target the hardest to hit and take the longest to select, the latter somewhere above 150 milliseconds and the former somewhere less that 75 milliseconds. The parameters of target jink interval and jink angle provide additional information that might explain results by Pavlovych and Gutwin [27] that showed more complex paths lowered target selection error rates.

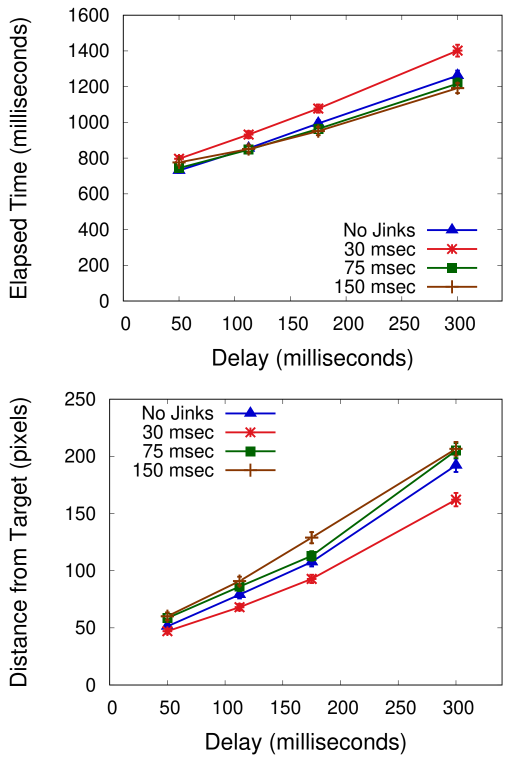


Figure 14: User selection time (top) and user accuracy (bottom) versus delay, data grouped by minimum target jink interval.

The user selection time (elapsed time) and user accuracy (distance) were each analyzed using a 3-factor ANOVA.[6] A summary of the results are shown in Table 3a and Table 3b, respectively. For both selection time and accuracy, there were significant main effects on delay, jink angle and jink interval, and significant interaction effects for delay-angle, and delay-interval. For selection time only, there was a significant interaction effect for angle-interval.

[6]Using R, https://www.r-project.org/

Factor	F value	p
delay	2192.44	< 2e-16
angle	321.63	< 2e-16
interval	219.91	< 2e-16
delay-angle	13.63	0.000223
delay-interval	70.02	< 2e-16
angle-interval	64.17	1.24e-15

(a) User selection time.

Factor	F value	p
delay	4435.363	<2e-16
angle	264.282	<2e-16
interval	404.660	<2e-16
delay-angle	150.044	<2e-16
delay-interval	89.256	<2e-16

(b) User accuracy.

Table 3: 3-factor ANOVA - delay, jink angle, jink interval.

For neither selection time nor accuracy was the interaction effect of delay-angle-interval significant.

4.3 Skill

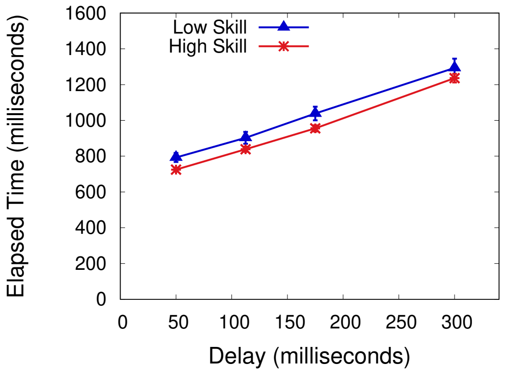
Selecting a rapidly moving target requires hand-eye coordination, so users that are skilled gamers more likely do better at Juke! – a potentially confounding effect. Users gave a self-rating of PC gamer skill (1 (low) to 5 (high)) from which we identified users with low skill (10 with rating 1-2) and high skill (28 with rating 4-5).

Figure 15 depicts the data analyzed by self-reported skill, with the axes as in Figure 10. For both graphs, there are two trendlines, one for each skill group. Each point is the mean for all users with that skill for that delay, shown with 95% confidence intervals. Overall, the decrease in performance as delay increases holds for both skill groups. With no added delay, there is little difference between the two skill groups – the 95% confidence intervals do not overlap, but the differences are only about 10%. For user selection time, as delay increases, the difference between skill groups stays the same (about 75 milliseconds). However, for user accuracy, as delay increases, there is a clear separation of the skill trendlines, with the high skilled users less affected by delays than the low skilled users. The low skilled users are about 20%-30% less accurate than the high skilled users in the presence of delays above 50 milliseconds.

4.4 Comparison with Other Studies

As mentioned in Section 2, previous work has examined the effects of delay on individual game actions, including moving target selection with a mouse. However, Claypool et al. [11] and Long and Gutwin [22] explored target selection with constant velocity targets while Juke! is the first to study force-based movement parameters for targets.

Figure 16 shows the Juke! user selection time data on the left graph, the PuckHunt game taken from Figure 5 of [11] (time to select target) in the middle graph, and the Pong game taken from Figure 2 of [22] (error rate in missing the target) on the right graph. All three games show similar trends in that user performance decreases with an increase in delay. However, both PuckHunt and



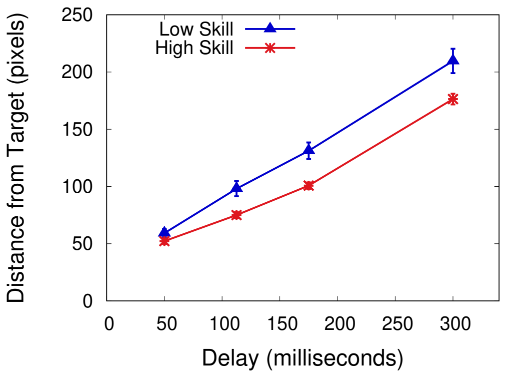


Figure 15: User selection time (top) and user accuracy (bottom) versus delay, data grouped by self-reported PC gaming skill.

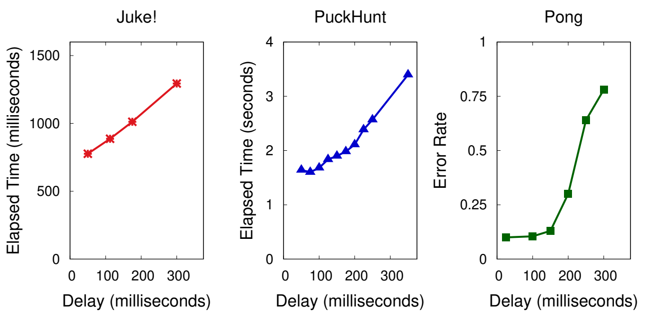


Figure 16: Comparison of Juke! with force-based target movement and PuckHunt and Pong with constant velocity target movement.

Pong with constant velocity target movement show a somewhat flat region on the left where performance does not degrade significantly for lower delays followed by a marked decrease in performance for delays above 200 milliseconds. This is in contrast to Juke! that has a steady, near linear, decrease in performance (increase in elapsed time) with an increase in delay over the range of delays tested.

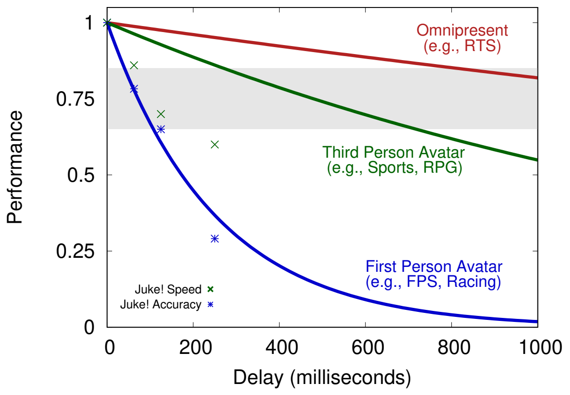


Figure 17: User performance versus delay for Juke! measurements and traditional network games.

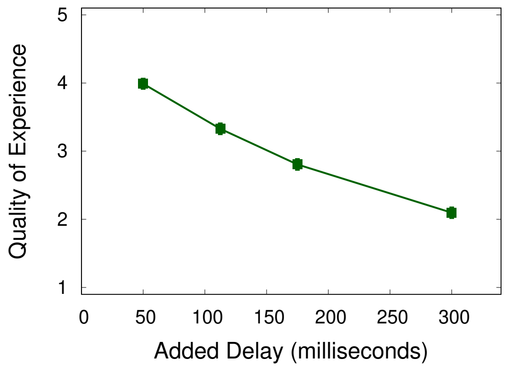


Figure 18: Quality of Experience versus delay.

4.5 Comparison with Traditional Network Games

To better understand how the Juke! measurements relate to a broader set of computer games, we follow earlier analysis [9] comparing the effects of delay on traditional network games. In general, the impact of delay differs depending upon how a user views the game world on a screen [10]: *first person* perspective where the user sees the game world through the eyes of the avatar, a *third person* perspective where the user follows an avatar in the virtual world, or an *omnipresent* perspective where the user has the ability to view and interact with different aspects of the game world. To allow for comparisons across different games, previous work [10] normalized and modeled earlier performance results ([3, 8, 13, 25, 26]) from 0 (worst) to 1 (best). We do the same for our user study data, normalizing the user selection time and accuracy (selection distance). In order to make our data comparable to the previously published results, the added base delay (50 milliseconds) is subtracted from our user study data.

Figure 17 depicts the results, summarizing classes of traditional network games in comparison to the Juke! user study results. The horizontal gray rectangle represents user tolerance for delay based on the previous studies. The exact user tolerance for delay depends on the game and to some extent the user, but gameplay quality is generally acceptable above the gray area and unacceptable below it. From the graph, the user selection time and accuracy in Juke! most closely follow the first person avatar perspective model. This is likely because performance in first person avatar games is most closely attuned to specific actions by the user (e.g., aiming a weapon) and so is directly impacted by delay, just as moving the mouse and clicking the mouse in Juke! is directly impacted by delay.

4.6 Quality of Experience

While user opinion of delay often correlates with performance [33], subjective measures can help ascertain the quality of the experience (QoE). For our Juke! user study, once for each jink interval, angle & delay combination, users were asked to rate the perceived amount of lag (1-low to 5-high) during the previous round (see Figure 6). These numbers are subtracted from 6 to align with typical QoE measures.

Figure 18 depicts a graph of the quality of experience versus delay. The x-axis is the total input delay and the y-axis is the reported lag perception (here, 1-high to 5-low). Each point is the mean rating for all users for that delay, shown with a 95% confidence interval. From the graph, there is an observable downward trend in QoE with an increase in delay, indicating users readily perceive added delays. Even with no added delay (far left), the mean of 4 suggests users perceive at least some local delay, while even at 300 milliseconds of delay (far right), users have not maxed out their perception of delay. Confidence intervals are small and non-overlapping in all cases, and mean values decrease about 50% (mean of 4 to mean of 2) from left (delay 50 milliseconds) to right (delay 300 milliseconds).

5 CONCLUSION

Computer games are increasingly networked, subjecting players to delays not only from the local system but also from the network and game servers, particularly for cloud-based games. This means player input, such as using a mouse to align a reticle on an opponent, is delayed between when the player moves the mouse until the game renders the change. While past work has shown delay degrades the player experience, details on the effects of delay on moving target selection, especially parameterized by target motion, have not been fully understood. In particular, what is needed is study of motion common to many games, where targets move along complicated movement paths, governed by turn frequency and turn angle with acceleration.

This paper presents the results of a study where users played a custom game called Juke! that isolated target selection and controlled for delay and target motion parameters – the time between target turns and the target turn angle range – using force-based motion. Fifty-six users played over 13,000 rounds of Juke! with the results analyzed along the independent variables of delay, jink angle and jink time, and with comparisons to previous work.

The analysis shows that both the distance between the mouse cursor and the moving target and the time it takes the user to select the moving target increase linearly over the range of delays tested (50-300 milliseconds), with user performance decreasing 4-fold over this delay range. The target jink angle has a lesser effect on distance (40% difference), with the hardest to hit target jink angle less than 360 degrees, and with the opposite effect on selection time. The target jink interval also has a lesser effect on distance (up to 30% difference) and selection time (up to 15% difference), with the hardest to hit target jink interval above 150 milliseconds. Skilled users are 20-30% less affected by delay than unskilled users.

Comparing the Juke! results with forced-based physics for target movement to previous results that use constant-velocity target movement shows somewhat similar trends in user performance degradation with increased delay, but a more noticeable degradation for Juke! at even low delays compared with the constant-velocity targets. These trends with delay generally match previous studies of traditional first person games with delay.

As noted in Section 4, in order to refine understanding of the effects of target turn angle and frequency, future work could provide for additional testing of more angles between 90 degrees and 360 degrees, as well as turn intervals greater than 150 milliseconds. Finding the threshold for tolerable delay for different movement parameters could also be of interest.

While our study controlled for target turn angle, turn interval and a maximum target speed, the acceleration force applied was constant. The magnitude of the force affects the target speed over time, particularly in the presence of turns. Future work could explore a range of forces, perhaps coupled with different maximum speeds.

Since another common game action is steering – navigating an avatar/vehicle down a virtual path – future work might study navigation in the presence of delay. In particular, such a future study might apply to a revised steering law [1], a predictive model of human movement that describes the time required to navigate, or steer, through a 2-dimensional tunnel.

Last, but not least, the 56 users sampled have an age range typical of university students. Future work could be to conduct user studies over a broader age range, perhaps considering additional demographic breadth as well.

ACKNOWLEDGMENTS

Thanks to Chaiwat Ekkaewnumchai and Bhon Bunnag for development of Juke! and conducting the user studies.

REFERENCES

[1] Accot, J., and Zhai, S. Beyond Fitts' Law: Models for Trajectory-based HCI Tasks. In *Proceedings of the ACM SIGCHI Conference* (Atlanta, GA, USA, 1997).

[2] Amin, R., Jackson, F., Gilbert, J. E., Martin, J., and Shaw, T. Assessing the Impact of Latency and Jitter on the Perceived Quality of Call of Duty Modern Warfare 2. In *HCI – Users and Contexts of Use* (Las Vegas, NV, USA, July 2013).

[3] Beigbeder, T., Coughlan, R., Lusher, C., Plunkett, J., Agu, E., and Claypool, M. The Effects of Loss and Latency on User Performance in Unreal Tournament 2003. In *Proceedings of ACM NetGames* (Portland, OG, USA, Sept. 2004).

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

[5] Buckland, M. *Programming Game AI by Example.* Jones & Bartlett, 2010.

[6] Chen, K., Haung, P., Wang, G., Huang, C., and Lee, C. On the Sensitivity of Online Game Playing Time to Network QoS. In *Proceedings of IEEE Infocom* (Barcelona, Spain, Apr. 2006).

[7] Chen, K.-T., Chang, Y.-C., Hsu, H.-J., Chen, D.-Y., Huang, C.-Y., and Hsu, C.-H. On the Quality of Service of Cloud Gaming Systems. *IEEE Transactions on Multimedia 26*, 2 (Feb. 2014).

[8] Claypool, M. The Effect of Latency on User Performance in Real-Time Strategy Games. *Elsevier Computer Networks, Special Issue on Networking Issues in Entertainment Computing 49*, 1 (Sept. 2005), 52–70.

[9] Claypool, M. Game Input with Delay - Moving Target Selection with a Game Controller Thumbstick. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) - Special Section on Delay-Sensitive Video Computing in the Cloud 14*, 3s (Aug. 2018).

[10] Claypool, M., and Claypool, K. Latency and Player Actions in Online Games. *Communications of the ACM 49*, 11 (Nov. 2006).

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

[12] Fitts, P. M. The Information Capacity of the Human Motor System in Controlling the Amplitude of Movement. *Journal of Experimental Psychology 47*, 6 (June 1954), 381–391.

[13] Fritsch, T., Ritter, H., and Schiller, J. H. The Effect of Latency and Network Limitations on MMORPGs: a Field Study of Everquest 2. In *Proceedings of ACM NetGames* (Hawthorne, NY, USA, Oct. 2005).

[14] Hajri, A. A., Fels, S., Miller, G., and Ilich, M. Moving Target Selection in 2D Graphical User Interfaces. In *Proceeding of IFIP TC Human-Computer Interaction (INTERACT)* (Lisbon, Portugal, Sept. 2011).

[15] Hoffmann, E. Capture of Moving Targets: A Modification of Fitts' Law. *Ergonomics 34*, 2 (1991), 211–220.

[16] Hoffmann, E. Fitts' Law with Transmission Delay. *Ergonomics 35*, 1 (1992), 37 – 48.

[17] Ivkovic, Z., Stavness, I., Gutwin, C., and Sutcliffe, S. Quantifying and Mitigating the Negative Effects of Local Latencies on Aiming in 3D Shooter Games. In *Proceedings of the ACM SIGCHI Conference* (Seoul, Korea, 2015).

[18] Jagacinski, R., Repperger, D., Ward, S., and Moran, M. A Test of Fitts' Law with Moving Targets. *The Journal of Human Factors and Ergonomics Society 22*, 2 (Apr. 1980), 225–233.

[19] Jarschel, M., Schlosser, D., Scheuring, S., and Hossfeld, T. An Evaluation of QoE in Cloud Gaming Based on Subjective Tests. In *Proccedings of the Conference on Innovative Mobile and Internet Services in Ubiquitous Computing* (Seoul, Korea, 2011), pp. 330–335.

[20] Jota, R., Ng, A., Dietz, P., and Wigdor, D. How Fast is Fast Enough?: A Study of the Effects of Latency in Direct-touch Pointing Tasks. In *Proceedings of the ACM SIGCHI Conference* (Paris, France, 2013).

[21] Kerr, R. Movement Time in an Underwater Environment. *Journal of Motor Behavior 5*, 3 (1973), 175 – 178.

[22] Long, M., and Gutwin, C. Characterizing and Modeling the Effects of Local Latency on Game Performance and Experience. In *Proceedings of the ACM CHI Play* (Melbourne, VC, Australia, 2018).

[23] MacKenzie, I. S., and Buxton, W. Extending Fitts' Law to Two-Dimensional Tasks. In *Proceedings of ACM SIGCHI Conference* (Monterey, CA, USA, May 1992).

[24] MacKenzie, I. S., and Ware, C. Lag As a Determinant of Human Performance in Interactive Systems. In *Proceedings of the ACM SIGCHI Conference* (1993).

[25] Nichols, J., and Claypool, M. The Effects of Latency on Online Madden NFL Football. In *Proceedings of the ACM NOSSDAV* (Kinsale, Ireland, June 2004).

[26] Pantel, L., and Wolf, L. C. On the Impact of Delay on Real-Time Multiplayer Games. In *Proceedings of the ACM NOSSDAV* (Miami, FL, USA, May 2002).

[27] Pavlovych, A., and Gutwin, C. Assessing Target Acquisition and Tracking Performance for Complex Moving Targets in the Presence of Latency and Jitter. In *Proceedings of Graphics Interface* (Toronto, ON, Canada, May 2012).

[28] Pavlovych, A., and Stuerzlinger, W. Target Following Performance in the Presence of Latency, Jitter, and Signal Dropouts. In *Proceedings of Graphics Interface* (St. John's, NL, Canada, May 2011), pp. 33–40.

[29] Quax, P., Monsieurs, P., Lamotte, W., Vleeschauwer, D. D., and Degrande, N. Objective and Subjective Evaluation of the Influence of Small Amounts of Delay and Jitter on a Recent First Person Shooter Game. In *Proceedings of ACM NetGames* (Aug. 2004).

[30] Raaen, K., and Eg, R. Instantaneous Human-Computer Interactions: Button Causes and Screen Effects. In *Proceedings of the 17th HCI International Conference* (Los Angeles, CA, USA, Aug. 2015).

[31] Raaen, K., and Petlund, A. How Much Delay Is There Really in Current Games? *Proceedings of ACM Multimedia Systems* (2015).

[32] Sackl, A., Schatz, R., Hossfeld, T., Metzger, F., Lister, D., and Irmer, R. QoE Management Made Uneasy: The Case of Cloud Gaming. In *IEEE Conference on Communications Workshops (ICC)* (Kuala Lumpur, Malaysia, May 2016).

[33] Saeed, S. S., Schmidt, S., Zadtootaghaj, S., Griwodz, C., and Moller, S. Towards Applying Game Adaptation to Decrease the Impact of Delay on Quality of Experience. In *Proceedings of IEEE International Symposium on Multimedia (ISM)* (Dec. 2018), pp. 114 – 121.

[34] Soukoreff, R. W., and MacKenzie, I. S. Towards a Standard for Pointing Device Evaluation – Perspectives on 27 Years of Fitts' Law Research in HCI. *Elsevier International Journal of Human-Computer Studies 61*, 6 (2004), 751 – 789.

[35] Ware, C., and Mikaelian, H. An Evaluation of an Eye Tracker as a Device for Computer Input. In *Proceedings of ACM Human Factors in Comp. Systems and Graphics Interface* (Toronto, Canada, Apr. 1987).