Game Input with Delay—Moving Target Selection with a Game Controller Thumbstick

MARK CLAYPOOL, Worcester Polytechnic Institute

Hosting interactive video-based services, such as computer games, in the Cloud poses particular challenges given user sensitivity to delay. A better understanding of the impact of delay on player-game interactions can help design cloud systems and games that accommodate delays inherent in cloud systems. Previous top-down studies of delay using full-featured games have helped understand the impact of delay, but often do not generalize or lend themselves to analytic modeling. Bottom-up studies isolating user input and delay can better generalize and be used in models, but have yet to be applied to cloud-hosted computer games. In order to better understand delay impact in cloud-hosted computer games, we conduct a large bottom-up user study centered on a fundamental game interaction—selecting a moving target with user input impeded by delay. Our work builds a custom game that controls both the target speed and input delay and has players select the target using a game controller analog thumbstick. Analysis of data from over 50 users shows target selection time exponentially increases with delay and target speed and is well-fit by an exponential model that includes a delay and target speed interaction term. A comparison with two previous studies, both using a mouse instead of a thumbstick, suggests the model's relationship between selection time, delay, and target speed holds more broadly, providing a foundation for a potential law explaining moving target selection with delay encountered in cloud-hosted games.

CCS Concepts: • **Human-centered computing** → **User models**; **User studies**; Laboratory experiments; • **Applied computing** → **Computer games**;

Additional Key Words and Phrases: Delay, lag

ACM Reference format:
Mark Claypool. 2018. Game Input with Delay—Moving Target Selection with a Game Controller Thumbstick. *ACM Trans. Multimedia Comput. Commun. Appl.* 14, 3s, Article 57 (June 2018), 22 pages.
https://doi.org/10.1145/3187288

1 INTRODUCTION

Cloud technologies that provide video-based services allow end-user devices to access full graphical content despite not having high-end graphics capabilities. While increasing network capacities may solve the substantive bitrate requirements, interactive cloud-based services have the additional challenge of needing to overcome delays that are inherent in cloud systems—most notably the round-trip delay required to send user actions to the server, process them, and send changes back to the client as video.

A promising type of interactive video-based service in the cloud is the computer game. Instead of distributing games through traditional media such as DVDs or electronic download, cloud-based

Author's address: M. Claypool, 100 Institute Road, Worcester, MA, 01609, USA.

https://doi.org/10.1145/3187288

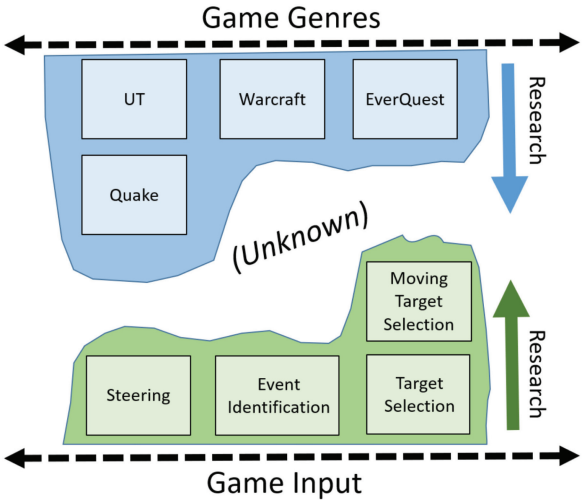

Fig. 1. Research in games and delay. Two approaches—top-down from existing games and bottom-up from user input.

games manage the game content itself on servers, sending down the game images as video to the client where the player reacts, then sends up the input to the servers [32]. Compared to traditional games, cloud-based systems give game providers more control over the content, while allowing users access to a large game library of graphics-intensive games using only a single, lightweight client.

Unlike for traditional network games where a client can potentially act on user input immediately, cloud-based games delay all user input by at least a round-trip time to the server [8]. Real-time games require players to make many time-sensitive actions that can suffer when the computer responses lag behind player input. Even delays as small as milliseconds can hamper the interplay between players' actions and intended results. For example, delay when aiming a virtual weapon can make it difficult for a player to hit a moving target in a shooting game, hurting the player's score and degrading the quality of experience.

While there are established methods to compensate for delays [5, 20], including system-level treatments (e.g., real-time priorities), delay compensation algorithms (e.g., dead reckoning, sticky targets, aim dragging), and even game designs to mitigate delay (e.g., delayed avatar response), an understanding of how delay affects fundamental player actions in games is needed in order to choose the most effective delay compensation techniques and, if possible, to develop and apply new techniques.

Broadly, there two different approaches to research in understanding the impact of delay on computer games, depicted in Figure 1. Studies using specific games are a top-down approach, extending knowledge of delay and games one game at a time. Such studies may try to generalize to game genres (e.g., first-person shooters) [2–4, 7, 9, 10], but game design and game engines may obfuscate important system details, making it difficult for measured results to pertain to other games and to analytic modeling. Moreover, most prior top-down studies have dealt with traditional computer games where user input may not always be delayed by the round-trip time to the server, unlike in cloud games where all input is delayed by the client-server round-trip time.

An alternate approach is bottom-up, studying fundamental game input (e.g., target selection), refined (e.g., moving target selection) and differentiated by hardware (e.g., mouse). Research con-

tributions to user input and delay has the potential to generalize to many games and even other interactive applications and allows for building analytic models that can explain and predict the effects of delay for a wide range of games and delay conditions.

While foundational studies of user input [15, 17, 22, 25] have shown promise in modeling user interaction for computer systems, including games [30], such studies have not focused on game actions (e.g., moving target selection with a game controller) nor have they focused on delay ranges present in cloud-based games. Ideally, game designers and system developers would have a model as far reaching and robust as Fitts' Law [15]—an ergonomic model for the time it takes for a user to select a target of a given size at a certain distance—but accurate for fundamental game actions in the presence of delays such as might be encountered in cloud-based video services. Our work takes a step toward developing such a law.

We design and implement a game that isolates the fundamental action of selecting a moving target with an analog thumbstick (such as is common on most game controllers) and controls the target speed and the delay between the user input and the rendered action. The game records the time it takes the user to select the target and gathers quality of experience ratings from the user. Applying a previously used methodology [12] to allow for data comparison, we deploy our custom game in a user study with over 50 participants, using delays ranging from 0 to 400 milliseconds and target speeds ranging from 150 to 450 pixels/second.

Analysis of the results shows the time to select a moving target with a thumbstick increases exponentially with delay—this is in contrast to earlier work [19] that showed a linear relationship when using a mouse. The time to select the target does not vary much with target speed for low delays, but there are significant interaction effects between added delays and target speeds for high delays—in other words, the time to select a fast target at high delay is larger than either parameter alone would suggest. User opinions on the quality of experience (responsiveness) show a pronounced linear decrease even for modest delay increases, confirming results in earlier work [6] and suggests even high-performance cloud gaming systems need to consider the impact of delays.

We derive an accurate analytic model for the average time to select a moving target with a thumbstick based on delay and target speed. The model is exponential with delay and includes an interaction term for delay and target speed. As a step toward a law describing moving target selection with delayed input, we compare the model with alternate models derived from two previous studies [11, 12] that had players use a mouse instead of a thumbstick. In doing so, we affirm that the generalized model produced for the thumbstick also holds for both mouse studies, albeit with different constants.

The rest of this article is organized as follows: Section 2 describes work on user input modeling related to our work; Section 3 describes our methodology, including developing our game and conducting a user study; Section 4 analyzes the user study results for overall trends and presents our model; Section 5 relates our results to earlier work and other systems; and Section 6 summarizes our conclusions and outlines possible future work.

2 RELATED WORK

This section presents related research in user input for target selection and delay.

2.1 Fitts' Law

Paul Fitts pioneered early seminal work in the area of human-computer interaction and ergonomics in the form of creating *Fitts' Law* [15]. With some simplification,[1] Fitts' Law describes

[1]To provide comparative clarity among proposed models, some constants and minor terms are ignored in this and subsequent models. The interested reader is encouraged to refer to the original sources for the full models.

the time (T) to select a stationary target based on an index of difficulty (I):

$$T = k \cdot I, \tag{1}$$

where k is a constant specific to the task at hand. The index of difficulty (I) is proportional to (1) the gap distance (G) from the source to the target, and (2) the width of the target (W):

$$I = \log_2 \left(\frac{G}{W} \right). \tag{2}$$

It has been suggested that target selection consists of a succession of discrete moves, each subdividing the distance to the target until the selector is within the target, similar to a binary search. In this light, the index of difficulty (I) can be interpreted as the average number of moves required.

Combining Equations (1) and (2), the time to select a target based on Fitts' Law is approximately

$$T = k \cdot \log_2 \left(\frac{G}{W} \right), \tag{3}$$

where G and W are known based on the task and the constant k is determined empirically through user study.

While Fitts developed and validated his law based on hand movements with a stylus, Fitts' Law has been shown to be applicable to a variety of other conditions (e.g., underwater [23]) and input devices (e.g., eye tracking [36]). And although subsequent adjustments to the model have shown a bit more predictive power by, for example, separating the width (W) and the gap distance (G) into separate terms with different constants [37], many newer models lose some of their intuition.

2.2 Fitts' Law with Two+ Dimensions

Fitts' Law only includes one dimension—the distance from the source to the target. To apply Fitts' Law to computer users selecting virtual targets with a mouse, subsequent enhancements examined the applicability of Fitts' Law to two dimensions [25], requiring a modest change in the "effective" width. For example, the W in Equation (3) is replaced by the smaller of the width and height for a rectangular target, although target shape was found to be largely irrelevant.

Since many modern uses of Fitts' Law are for computer devices with two dimensional displays, MacKenzie and Buxton's [34] investigation of Fitts' Law provided guidelines for use of the law in evaluating pointing devices.

For higher dimensions, such as for 3d target selection in a virtual environment, Murata and Iwase [27] found that conventional Fitts' did not match user performance well owing to a missing directional component, and So and Chung [33] found target width and target distance interacted differently with display delay (i.e., in the virtual headset) and controller delay.

2.3 Fitts' Law with Moving Targets

Fitts' Law also only applies to stationary targets. This is entirely appropriate when used for, say, a typical PC application where a user is selecting a stationary user interface button with the mouse. However, stationary targets are less appropriate for a real-time dynamic interface, such as a computer game, where targets are often changing positions on the screen.

Jagacinski et al. [21] extended Fitts' Law with a revised index of difficulty (I, see Equation (2)) that explicitly incorporates target velocity and which predicts the overall pattern of target selection times better than the original index of difficulty. Their revised model for the time (T) to select a moving target at gap distance (G) based on the target's speed (S) and width (W) is

$$T = k_1 \cdot G + k_2 \cdot \frac{S}{W}, \tag{4}$$

where k_1 and k_2 are constants determined empirically.[2] Note, Jagacinski's model suggests target selection time increases linearly with target speed.

User study experiments were done with an oscilloscope, where eight users moved a center line to be between two horizontal lines. There were two different kinds of joystick controls—positional and mouse. Target velocities were 0, 30, and 60 millimeters/second, always toward the center.

Hoffmann [18] refined Jagacinski's model:

$$T = k \cdot \log_2 \left(\frac{G + S}{W - S} \right), \tag{5}$$

where S, G, and W are as for Jagacinski and k is an empirically derived constant. Note, although it is not immediately obvious, Hoffman's model suggests target selection time increases exponentially with target speed, effectively $log_2(\frac{1}{1-S})$. While Hoffman showed his new model fit the data from Jagacinski's experiments somewhat better, the model loses some intuition and simplicity when compared to Jagacinski's model.

Hajri et al. [17] refined Hoffmann's model by separating the index of difficulty from Fitts' (I, see Equation (2)) into one that accounts for target speed:

$$I = \log_2 \left(\frac{G \pm \frac{S}{k}}{\frac{W}{2} - \frac{S}{k}} \right), \tag{6}$$

where k is, again, an empirically derived constant. Note, the ± generalizes from + in Hoffmann's model since the target may be moving toward (i.e., −) or away from (i.e., +) the source. A user study showed Hajri's model fit the experimental data well.

2.4 Fitts' Law with Delay

The above models have assumed a constant, low delay—i.e., a system with only the delay inherent in the local electronics and software. However, in many modern systems (e.g., client-server systems) there are not only delays from the local computer system but also from network transmissions and processing on a remote computer system (e.g., a server). This is particularly true for cloud-hosted games where a cloud-based server must receive and process all player actions before rendering the scene and sending it back to the client to be displayed.

Suitable for this environment, Hoffman [19] revised Fitts' Law with delay:

$$T = k_1 + k_2 \cdot D \cdot \log_2 \left(\frac{G}{W} \right), \tag{7}$$

where D is the delay, G and W are the target gap distance and width, respectively, and k_1 and k_2 are empirically derived constants. Hoffmann's model shows a multiplicative effect between the index of difficulty and the delay and suggests target selection time increases linearly with delay.

Hoffman's experiments using one-dimensional input (a knob to move a pen from the source to the target) shows 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; and (2) *continuous* where a user provides continuous input in the presence of delay, adjusting as necessary without stopping. For low delays, users generally provide continuous input while for high delays, users provide move-and-wait input. Hoffman's experiments found the delay inflection point between continuous and move and wait to be around 700 milliseconds.

MacKenzie and Ware [26] proposed the same model as Hoffman, and did an eight-person user study that measured the effects of delay on target selection time using a computer mouse for added

[2]Jagacinski separated the gap distance (G) from the target width (W), much as did other early enhancements to Fitts' Law.

delays up to 225 milliseconds. Their experiments explained up to 94% of the variation in average selection times.

Ware and Balakrishnan [35] applied Hoffman's model to a 3d selection task (a virtual fish tank), noting that the delay (D in Equation (7)) is both from the device lag (e.g., the time between the user moving the mouse and the application recording this movement) and the display lag (e.g., the time between updating a mouse location and the change being displayed on the screen).

Jota et al. [22] studied the impact of delay on selection and then dragging with touch devices, comparing experimental results to MacKenzie and Ware's [26] Fitts' Law with delay. User study results with 20 participants confirmed that Fitts' Law holds when determining different constants for each level of delay.

Brady [6] experimented with target selection and delay, considering index of difficulty as in Equation (2) for delay (including frame rate delays), but not deriving experimental parameters for Hoffman's model (Equation (7)). Brady also analyzed the subjective user experience, finding a strong inverse linear relationship over the range of delays studied (from 0 to about 200 milliseconds).

2.5 Missing—Fitts' Law with Transmission Delay and Moving Targets

What is missing is analysis and, where appropriate, models that explain and predict the time to select a *moving target with delay*, specifically with an analog game controller thumbstick. Our work takes a step to address this deficiency.

3 METHODOLOGY

To model moving target selection and delay, we (1) develop a game (*Puck Hunt*) that enables study of user input with controlled delay (Section 3.1); (2) conduct a user study to evaluate the impact of a single user action with delay (Section 3.2), (3) analyze the core results of the user study through graphs (Section 4), (4) derive an explanatory model (Section 4.3) and compare it to previous studies (Section 4.4), and (5) analyze other results from the user study, particularly as compared to traditional and cloud-based games (Sections 4.5 to Section 4.9).

3.1 Game

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 *Puck Hunt*, named after and based on the classic game *Duck Hunt* (Nintendo, 1984), depicted in Figure 2. Puck Hunt allows study of a single user action with controlled amounts of delay. In Puck Hunt, depicted in Figure 3, the user proceeds through a series of short rounds, where each round has a large black ball, the puck/target, that bounces around the screen using predictable physics. The user moves the thumbstick to control the small red ball (a.k.a., the cursor) and attempts to select the target by moving the ball over the target and pulling the controller trigger. Once the user has successfully selected the target, the target disappears and a notification pops up telling the user to prepare for the next round. Thereupon pressing the controller "A" button, a new round starts, with the target at a new starting location with a new direction and new speed. The user is scored via a timer that counts up from zero at the beginning of each round, stopping when the target is selected.

The action chosen—selection of a 2d moving target—is common to many game genres, such as first-person shooters (FPS) (e.g., *Call of Duty*, Activision, 2003) and multiplayer online battle arenas (MOBAs) (e.g., *League of Legends*, Riot Games, 2009).

Fig. 2. *Duck Hunt* (Nintendo, 1984) screenshot. Users move the cursor (circle with the line through it) using the game controller and click on the moving duck.

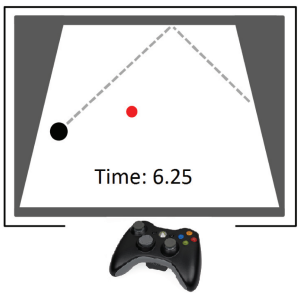

Fig. 3. Puck Hunt. Users move the cursor (red ball) with a thumbstick and click on the moving target (puck). The game adds delay to user input (both movement and clicking) and varies target speeds between each round.

Table 1. Target Speeds and Added Input Delays Used in User Study

Speed (pixels/second)	
Slow	150
Medium	300
Fast	450

(a) Target speeds.

Delay (milliseconds)
0, 25, 50, 75
100, 125, 150, 175
200, 300, 400

(b) Added input delays.

Puck Hunt is written in C++ using OpenGL to minimize inherent software delay, with support from the Angel 2d game engine.[3] Puck Hunt runs in fullscreen mode at 1080p resolution (1920×1080 pixels) and 60f/s. The target is 100 pixels in diameter and the cursor (the red ball) is 25 pixels in diameter.

Each round, the target moves with one of three possible speeds listed in Table 1(a). Effectively, these speeds create levels of difficulty that might be experienced in 2d shooting games (e.g., *Duck Hunt* (Nintendo, 1984)) or third-person combat games (e.g., *League of Legends* (Riot, 2009)). The game adds a controlled amount of delay selected from the set in Table 1(b). The delay is added to all user input (thumbstick movement and trigger pulls) for the duration of the round. The set of delays is chosen so as to explore in detail delays up to 200 ms (common in many broadband networks and systems), while allowing some exposure to larger delays (common in some wireless and wide-area networks). Each delay and speed combination appears five times, but the entire set of combinations is shuffled into a different random order for each user.

Exactly once for each combination of delay and speed, the user is asked to rate the quality of experience (QoE) based on the responsiveness during the round, shown in Figure 4. The game pauses until the user selects a choice, 1 (low) to 5 (high).

[3] http://angel2d.com/.

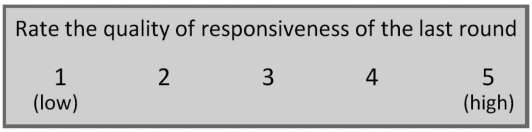

Fig. 4. Quality of experience prompt to player.

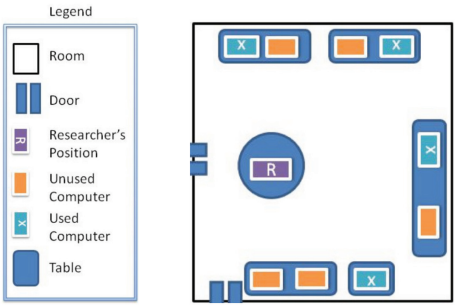

Fig. 5. Lab for user study.

Every 30 rounds, the game stops for a minimum of 20 seconds to allow the user to rest/regain concentration, with a countdown timer shown to the user via a popup window.

3.2 User Study

Our user study was conducted in a windowless computer lab with bright, fluorescent lighting, with the layout shown in Figure 5. The computers were Dell PCs with Intel i7-4790 4GHz processors, 4GB NVidia GeForce GTX 960 graphics cards, and 16GB of RAM, running Microsoft Windows 7. The monitors were 24" Dell U2412M LCDs with a native resolution of 1920×1200 pixels and a refresh rate of 59p Hz.

Participants were volunteers solicited through WPI email lists and the Social Science Research Participation System http://wpi.sona-systems.com/, a system whereby Psychology students participate in user studies to obtain class and major credit. Added incentives included a raffle for a $25 gift card for participating and a $25 gift card for the user with the highest score. Game development students that participated also received one extra point on their final exams.

First, users heard a scripted brief about the study and signed an Institute Review Board (IRB) consent form at the researcher's position (noted in Figure 5). Next, users were asked to make themselves comfortable at a computer by adjusting chair height and monitor angle/tilt so as to be looking at the center of the screen. Users were encouraged to use the controller with whichever hand they preferred.

Users logged into the computer using their WPI credentials and were asked to open a Web browser with a survey coded using the Qualtrics survey tool.[4] Users then completed a survey about demographics and gaming experience. The game and incentive options were then described followed by launching the game.

Play commenced immediately, but the first two rounds with no added delay and slow targets were used for practice only and the results were not recorded. Play then proceeded through all 5× shuffled combinations of delay and speed (Table 1(a) and Table 1(b)), with one QoE question for

[4]https://www.qualtrics.com/.

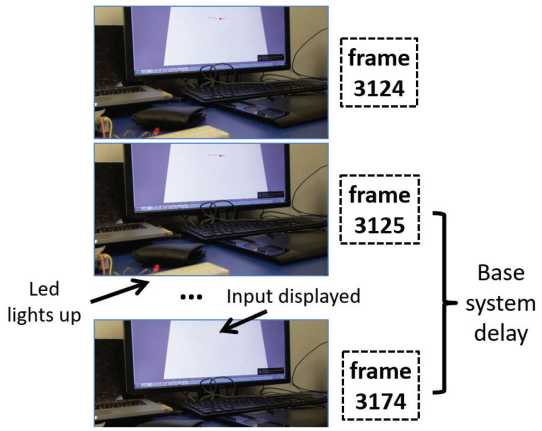

Fig. 6. Measuring base delay.

each delay and speed combination and a forced pause every 30 rounds. In total, users played 165 recorded rounds each, which took about 15 minutes including answering questions and pausing.

3.3 Local System (Base) Delay

Note, the delays in Table 1(b) added by Puck Hunt are in addition to any delays inherent in the base computer system. Since such base delays have been shown to be significant [20, 31], we measured the base delay for game controller actions on our lab computers using a Blur-busters type technique.[5]

A breadboard with an LED was connected via a wire soldered to a game controller so that the LED lit up when the controller trigger was clicked (user pulled the controller trigger). A high frame rate camera (a Casio EX-ZR200) filmed the player selecting the QoE prompt using the trigger, recording the action at 1,000f/s. By manually examining the individual video frames, the frame number when the light appeared when the trigger was clicked is subtracted from the frame number when the QoE prompt showed the input, giving the base delay.

Figure 6 depicts an example of the measurement method. The cursor is poised over the QoE prompt in frame 3,124. In frame 3,125, the trigger has been clicked indicated by the lit LED on the breadboard. The input is not displayed on the QoE prompt until frame 3,174. Since there is one video frame each millisecond, subtracting 3,125 from 3,174 gives a base delay of 49 milliseconds.

The measurement method was repeated four times, resulting in base delay values of 46, 54, 50, and 49 milliseconds. Hence, 50 milliseconds is added to all delay analysis.

4 ANALYSIS

This section starts with demographics (Section 4.1), then presents the core results of target selection time measurements (Section 4.2) and target selection time models (Section 4.3 and Section 4.4). Further analysis breaks down target selection times by player skill (Section 4.5), analyzes trigger pull (click) measurements (Section 4.6), and compares the results with traditional network games (Section 4.7) and other cloud game studies (Section 4.8). The section ends with a brief QoE analysis (Section 4.9).

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

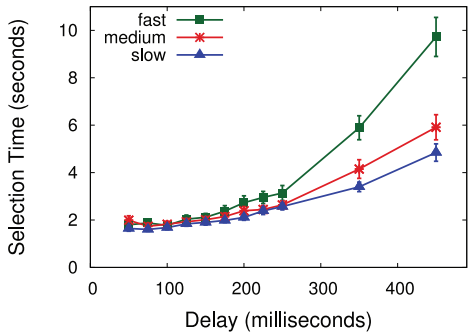

Fig. 7. Mean selection time versus delay; data grouped by target speed.

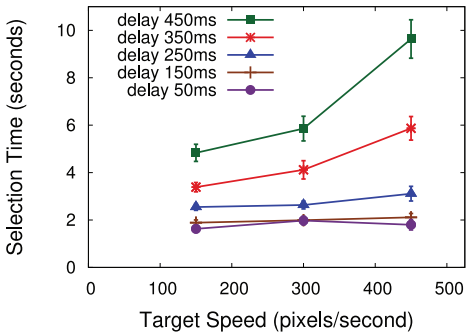

Fig. 8. Mean selection time versus target speed; data grouped by delay.

4.1 Demographics

Fifty-one users participated in the study. Ages ranged from 18 to 25 years with a mean and median of 20 years. Thirty-two identified as male, 16 as female, and 3 did not specify. Forty-four indicated they were right-handed, 6 left-handed, and 1 ambidextrous.[6] The mean self-rating as a PC gamer (scale 1–5) was about 3.5, showing a slight skew to having "high ability." About half the users played 6+ hours of computer games per week. Most studied Computer Science, Game Development, or Engineering.

4.2 Target Selection Time—Measurement

We first assess the time it takes for the player to select a moving target with a thumbstick in the presence of delay.

Users were able to select the target at all speeds and all delays within 30 seconds (the maximum time allowed) with the exception of the fastest targets and highest delays, where 5% of the time users were unable to select the target within the time limit. In order to obtain a representative average not skewed by the time ceiling, we model the select time cumulative distribution functions (CDFs) of the fastest target and highest delay using only scores below 30 seconds. This way, the model can be extended past the 30 second time limit to provide for a full distribution range and allowing for derivation of a mean selection time. Because CDFs naturally trend to 1, we fit[7] a logarithmic regression model to data points below 30 seconds and then integrated to yield the means. Doing so provides an analytic model of $y = 0.36 \cdot ln(x) - 0.20$ (R^2 0.98) and a mean of 9.80 milliseconds. This mean value is used in all subsequent analyses for the fastest targets and highest delays.

Figure 7 depicts selection time versus delay, grouped by target speed. The x-axis is the total input delay (added delay+base delay) and the y-axis is the time to select the moving target. There are three trendlines, one for each target speed tested. Each point is the mean time for all users for that delay and speed combination, shown with a 95% confidence interval. Overall, there is an increase in mean selection time as delay increases. This increase appears exponential over the range of delays tested. For delays under 200 milliseconds, the speed of the target does not significantly impact mean selection time. However, starting at delays of 250 milliseconds, the faster speed targets become significantly harder to select than the slowest speed targets. At the extreme

[6]We did not observe which thumb users used to manipulate the thumbstick.
[7]Using R, https://www.r-project.org/.

Table 2. User Study Data Values Used for Standardization

Factor	Mean	Std. Dev.
Delay	191 milliseconds	115
Speed	300 pixels/second	122

delay (450 milliseconds), the fastest targets take 2.5× longer to select than when there is minimal delay (100 milliseconds) and even the slowest targets take over 2× longer to select.

Figure 8 graphs the same data as in Figure 7, but grouped by delay and analyzed by speed. The y-axis is the same, but the x-axis is target speed in pixels per second. There are five trendlines, one for each of the total input delays (added delay+base delay).[8] Each point is the mean time for all users for that delay and speed combination, shown with a 95% confidence interval. Overall, there is an increase in selection time as the target speed increases. This increase appears mostly linear for the range of target speeds tested, but is somewhat non-linear (perhaps exponential) for delays above 300 milliseconds. Delay impacts the selection time for all target speeds, but is most pronounced for the highest target speeds as seen by the diverging lines. For delays of 200 milliseconds and under, the lines in Figure 8 are flat—the speed of the target does not impact mean target selection time.

4.3 Target Selection Time—Model

While trends in user performance with delay provide valuable insights, an analytic model illustrating the relationships can be used by developers and researchers for designing and implementing more effective interactive systems and applications in the presence of delay. Hence, we model the mean time to select a moving target with a thumbstick with delayed input.

Based on the previous analysis (e.g., see Figure 8), (1) there is a clear upward trend in mean time to select a moving target with increased delay and with increased speed, and (2) the greater increase in time to select a moving target at fast speeds and high delays suggests considering the interactions between speed and delay. Thus, we explore models of the time to select a moving target with a thumbstick (T) with possible terms for delay (D), speed (S), and combined interaction terms.

To build our models, we standardized our user study data[9] using the values in Table 2. This allowed us to fit regression models using R.[10]

There are many possible models that fit our experimental data. In order to compare different regression models, the coefficient of determination (R^2) can be used as a measure of how well observed outcomes are replicated by the model based on the proportion of total variation of outcomes explained by the model. In doing such a comparison, it might be tempting to choose the model with the maximum R^2, but this can over-fit the model to the data (i.e., the R^2 value can be increased by adding more terms). However, the adjusted R^2 modifies the R^2 based on the number of predictors in the model, increasing if the new term improves the model more than would be expected by chance and decreasing it otherwise. Overall, we want a *parsimonious* model—one that provides the desired prediction with as few terms as possible.

Table 3 summarizes the models explored, ordered in increasing adjusted R^2 (higher is better). For the equations, the k parameters (e.g., k_1) are constants, e is the exponential function, d is delay, and s is speed.

[8]The other delays tested are not shown to keep the graph readable.
[9]Subtracting the mean and dividing by the standard deviation.
[10]https://www.r-project.org/.

Table 3. Models Predicting Selection Time of a Moving Target With Delay (d) and Speed (s)

	Model	Adj. R^2
1.	$k_1 + k_2 s$	0.035
2.	$k_1 + k_2 e^s$	0.038
3.	$k_1 + k_2 d$	0.696
4.	$k_1 + k_2 d + k_3 s$	0.745
5.	$k_1 + k_2 e^d$	0.781
6.	$k_1 + k_2 e^d + k_3 e^s$	0.835
7.	$k_1 + k_2 d + k_3 d^2 + k_4 s + k_5 s^2$	0.840
8.	$k_1 + k_2 d + k_3 s + k_4 ds$	0.844
9.	$\mathbf{k_1 + k_2 e^d + k_3 e^s + k_4 e^d e^s}$	**0.981**
10.	$k_1 + k_2 d + k_3 d^2 + k_4 s + k_5 s^2 + k_6 ds + k_7 d^2 s + k_8 ds^2 + k_9 d^2 s^2$	0.996

The letter e is the exponential function, and letters with k are constants (different for each model), derived from data gathered through users study experiments.

While model #10 has the highest adjusted R^2, it is only slightly higher than model #9 while having significantly more terms (nine versus four). Moreover, previous work has suggested an exponential relationship with speed in the time to select a moving target without delay [17, 18]. Thus, we propose modeling the time to select a moving target with a thumbstick (T) with exponential functions for delay (d) and speed (s), as well as an interaction term:

$$T = k_1 + k_2 e^d + k_3 e^s + k_4 e^d e^s, \tag{8}$$

where k_1, k_2, k_3, and k_4 are constants derived from data gathered through user study experiments.

Using our standardized user study data[11] in Table 2 yields an adjusted R^2 0.981, F-stat 547, and $p < 2.2 \times 10^{-16}$. This simplified final model for the time to select a moving target with a delayed thumbstick (T) is

$$T = 2 + 0.3e^d - 0.03e^s + 0.2e^d e^s, \tag{9}$$

where d and s are standardized from the total delay (D) and target speed (S), respectively: $d = \frac{D-191}{115}$ and $s = \frac{S-300}{122}$.

All terms are significant at $p < 0.001$, except the main effect of speed ($0.03e^s$ with $p < 0.38$).

4.4 Target Selection Time—Model Comparison

As described in Section 1, a *scientific law* is a statement based on repeated experimental observations that accurately describes an aspect of the world. Specific to our context, a desired outcome would be a law that explains the time for a user to select a moving object with a pointing device (e.g., a thumbstick or a mouse) when the input is subject to delay.

With this in mind, we compare the model produced in Section 4.3 with models from two previous studies, named WPI-M [11] and WO-M [12], that analyzed data on moving target selection with delayed input. Both WPI-M and WO-M used a methodology similar to Section 3, but with the significant difference of using a mouse instead of a game controller thumbstick for target selection. The studies had completely different users and different sample sizes, with the users of WO-M drawn from the population of students at Westerdals University in Olso, while users in WPI-M and WPI-T (our study) were drawn from the population of Worcester Polytechnic Institute in the

[11] Subtracting the mean and dividing by the standard deviation.

Table 4. Differences in Study Parameters for Three Studies

Name	Sample Pool	Selection Device	Speed (pixels/second) Slow	Med.	Fast	Base Delay (msec.)
WO-M	Norway Univ.	mouse	550	1,100	1550	20
WPI-M	U.S. Univ.	mouse	150	300	450	100
WPI-T	U.S. Univ.	thumbstick	150	300	450	49

Any parameters not shown are the same across all studies.

Table 5. Summary of Key Demographics

Name	Mean Age (yrs)	Gender	Device Hand	Self-Reported Mean Ability PC Game
WO-M	23.9 (3.5)	45 ♂, 8 ♀	52 right, 1 left	3.8 (1.0)
WPI-M	20.9 (1.9)	23 ♂, 8 ♀, 1 ?	32 right	3.5 (1.3)
WPI-T	20.0 (1.6)	32 ♂, 16 ♀, 3 ?	44 right, 6 left, 1 ambi	3.5 (1.2)

Standard deviations for mean values are shown in parentheses. Mean ability is from 1 (low) to 5 (high).

Table 6. Comparison of the Three Models

Name	Selection Time Model (seconds)	d (msec.)	s (pixels/s)	Adj. R^2	F-stat
WO-M	$T = 1 - 0.08e^d - 0.6e^s + 2.3e^d e^s$	(D-165)/115	(S-1067)/409	0.99	3,352
WPI-M	$T = 1 + 0.2e^d - 0.04e^s + 0.1e^d e^s$	(D-245)/114	(S-300)/122	0.97	328
WPI-T	$T = 2 + 0.3e^d - 0.03e^s + 0.2e^d e^s$	(D-191)/115	(S-300)/122	0.98	547

U.S. WO-M also had significantly different target speeds where all three target speeds were higher than those in WPI-M and WPI-T. Lastly, all three studies had different base delays since they ran on different computer systems. Table 4 summarizes the study parameters that differ across WPI-M, WPI-T, and WO-M. Any parameters not listed are the same as in Section 3 for all three studies.

The user demographics differed somewhat for each study, across age, gender, and skill breakdown, as summarized by Table 5. Generally, each user sample consisted of mostly male, undergraduate gamers.

Following similar reasoning as in Section 4, we proposed the general model in Equation (8) for relating target selection time to speed and delay for WPI-M and WO-M. Table 6 compares the three models. For all models, $p < 2.2 \cdot 10^{-16}$ and all terms are significant at $p < 0.001$, except the main effect of speed (the third term, $k_3 e^s$). Overall, the proposed general equation fits every study well based on the high adjusted R^2 values.

Figure 9 depicts the models for a speed of 550 pixels/second versus delay, each charted over the domain of delays tested. From the figure, the models agree in shape, with a marked increase in selection times with an increase in delay, particularly for delays above 300 milliseconds. The WPI-T and WPI-M curvatures are similar, with the WPI-T curve shifted higher by approximately 1 second—this "extra" second may be attributed to the average increase in difficulty of using a thumbstick versus a mouse.

For illustration, Figure 10 depicts the models for a delay of 250 milliseconds, each charted over the domain of speeds tested. From the figure, the curves for WPI-T and WPI-M are similar, only increasing slightly over the range of speeds tested. As for the previous graph, WPI-T is shifted up compared to WPI-M by about a second. In contrast, WO-M has a noticeable increase in selection

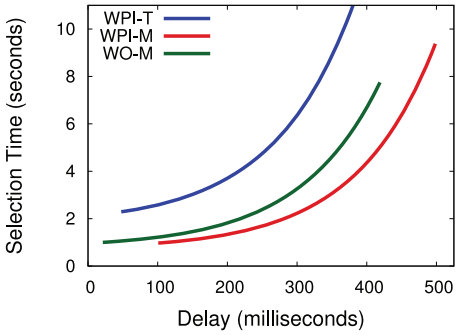

Fig. 9. Model of mean selection time versus delay; speed 500 pixels/s.

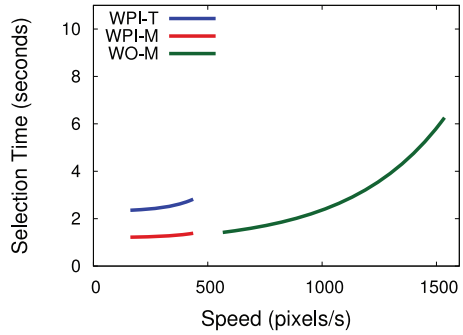

Fig. 10. Model of mean selection time versus speed; delay 250 milliseconds.

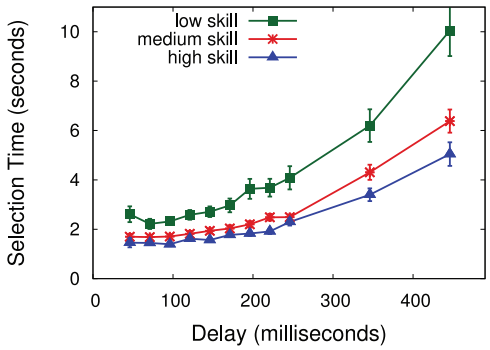

Fig. 11. Mean selection time versus delay, grouped by skill.

time with the much higher speeds it models. The WPI-M and WO-M appear to be on a similar trajectory from 450 to 525 milliseconds.

In summary, the accuracy of the same model construction for three independent studies suggests (a) target selection time is exponential with delay, (b) target selection time may be exponential with target speed, but such terms are not significant (at $p < 0.001$) for any model, and (c) there is an interaction between delay and target speed that is not captured by either term alone.

4.5 Target Selection Time by Skill—Measurement

Rapid moving target selection requires dexterous hand-eye coordination. Thus, one confounding effect to any model of target selection is the skill of the user. Users that are more skilled at computer games (where moving target selection is common) are likely more skilled in our target selection task.

For our study, users provided a self-rating of gamer skill, from 1 (low) to 5 (high). Based on our user sample, we triaged users into low skill (12 users with rating 1–2), medium skill (26 users with rating 3–4), and high skill (13 users with rating 5).

Figure 11 depicts selection time versus delay for the fast targets only,[12] analyzed by self-reported skill. The axes are as for Figure 7. There are three trendlines, one for each skill group. Each point is the mean selection time for all users with that skill, analyzing only fast speed targets for that

[12]Slower targets showed the same trends, albeit with less of a distinction between trendlines.

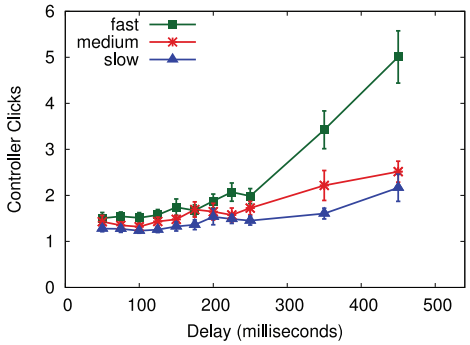

Fig. 12. Game controller clicks versus delay, grouped by target speed.

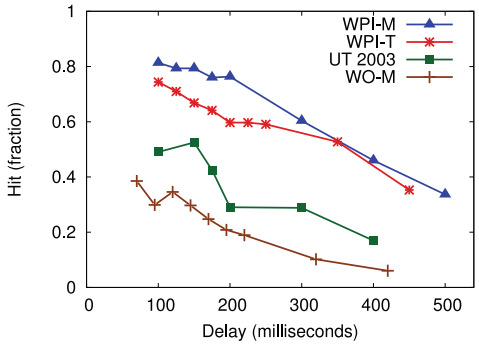

Fig. 13. Hit fraction versus delay, grouped by game/model.

delay. The means are shown with 95% confidence intervals. Overall, the increase in mean selection time as delay increases holds for all skill groups. For delays under about 250 milliseconds, there is only a slight separation between the medium and the high skill groups. However, there is clearer separation of the skill trendlines for delays of 350 and 450 milliseconds. At these delays, the most skilled users are affected by delays the least and the least skilled users the most. At the extreme delay (450 milliseconds), the least skilled users take about 2× longer to select the target than the most skilled users.

4.6 Controller Clicks—Measurement

Another problem reported by some Fitts-type user studies is that an "aggressive" user repeatedly clicking the selection button as rapidly as possible can select the target faster, albeit with more clicks that "miss." For many computer games, missing a target can have a cost (i.e., wasting virtual ammunition) and so can also be used as a measure of performance. In our case, all clicks above one per target have missed the target.

Figure 12 depicts clicks (users pulling the controller trigger) versus delay, analyzed by target speed. The x-axis is the total input delay and the y-axis is the number of clicks needed to select the moving target. There are three trendlines, one for each target speed tested. Each point is the mean number of clicks for all users for that delay and speed combination, shown with a 95% confidence interval. Overall, there is an increase in mean number of clicks as delay increases. For fast targets, clicks increase exponentially over the range of delays tested somewhat the same as does selection time, but for slow and medium targets, the increase appears more linear over the range of delays tested. For delays under 200 milliseconds, the speed of the target does not impact the number of clicks. At the extreme delay (450 milliseconds), users miss about 250% more often for fast targets as opposed to slow targets.

Since player performance in cloud-based games is a major focus of the proposed research, we compare the clicking performance of the users in Puck Hunt to shooting performance in a commercial first-person shooter game. A previous study [4] with *Unreal Tournament (UT) 2003* (Atari, 2002) recorded the hit fraction versus delay, where users tried to hit a moving avatar with a high precision weapon using a mouse as a selection device. Hits plus misses in UT 2003 is analogous to total clicks in Puck Hunt, and the Puck Hunt hit fraction is 1 divided by the number of clicks. Figure 13 compares the results of UT 2003 with Puck Hunt. The x-axis is the total input delay and the y-axis is the hit fraction. There are four trendlines, one for UT 2003 and one for each Puck Hunt study, with each point the average across all target speeds. Overall, the trendlines follow the

same pattern, decreasing approximately linearly with an increase in delay. The WPI Puck Hunt hit fractions are significantly higher than the UT 2003 hit fractions, while the WO-M Puck Hunt hit fractions are significantly lower than the UT 2003 hit fractions. This trend may be entirely due to target speed, but, in the case of the WPI studies, may also be because the UT 2003 avatar was controlled by a human and moved unpredictably, while the Puck Hunt target moved with constant velocity. The WPI-M hit fraction is slightly higher than the WPI-T fraction, which is perhaps expected as the mean selection times with the mouse are slightly lower than the mean selection times with the thumbstick.

4.7 Comparison with Traditional Computer Games

To better understand how the Puck Hunt measurements of selection time with delay relate to a broader set of computer games, we turn to previous work on the effects of delay on computer games [10]. In general, the impact of delay differs depending upon the game perspective—i.e., how a user views the game world on a screen.

With an avatar-interaction perspective, the user interacts with the game through a single representative character, called the *avatar*. Games with an avatar-interaction typically have either a *first-person* perspective where the user sees the game world through the eyes of the avatar, or a *third-person* perspective where the user follows an avatar in the virtual world. FPS games, role-playing games (RPGs), sports games, and racing games are all examples of game genres that have an avatar-interaction perspective. These game genres often differ in the perspective—for example, FPS games have a first-person perspective while RPGs typically have a third-person perspective.

With an *omnipresent* perspective, the user has the ability to view and interact with different aspects of the game world. The user is said to be omnipresent in that his/her actions have a more global influence than actions in an avatar model. The perspective of games with the omnipresent interaction model is often variable, giving users an aerial perspective to provide a bird's eye view of the virtual world, but also allowing users to zoom in to a third-person perspective to provide fine grained control over individual resources. Real-time strategy (RTS) games and construction and simulation games are examples of game genres with the omnipresent perspective.

In order to compare the effects of delay across games, objective user performance results are normalized from 0 (worst) to 1 (best). Results from previous studies[13] of delay and traditional network games (first-person avatar [4, 29], third-person avatar [16, 28], and omnipresent [9]) are similarly normalized and fit with an exponential curve [10]. The same is done for our user study data, normalizing the selection time based on target speed. 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 for this graph.

Figure 14 depicts the results, summarizing classes of traditional network games. The horizontal gray rectangle is a visual indicator of user tolerance for delay. Gameplay quality is generally acceptable above the gray area and unacceptable below it. The exact delay tolerance threshold depends on the game and to some extent the user's own perception and sense of immersion (hence the gray color and the rectangle shape rather than a line). From the graph, the time to select a moving target with a thumbstick most closely follows the first-person avatar perspective model. This is likely because performance in such games is most closely attuned to specific actions by the user (e.g., aiming a weapon) and so is severely impacted by delay, whereas in other game models user performance is handled by virtual agents with some autonomy so the effects of delay are mitigated.

[13]Providing details on these studies is too cumbersome for this article, but the interested reader is encouraged to follow the references.

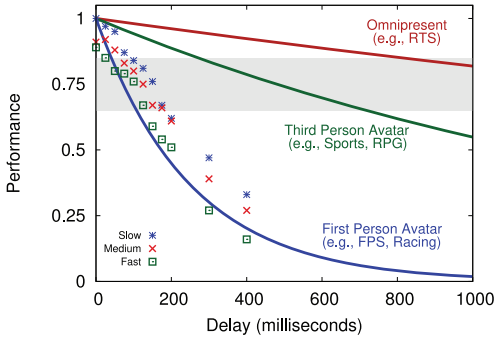

Fig. 14. User performance versus delay for Puck Hunt measurements and commercial game models.

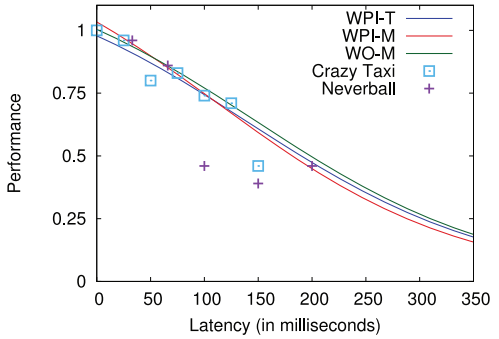

Fig. 15. User performance versus delay for Puck Hunt models and cloud game measurements.

4.8 Comparison with Cloud-Based Computer Games

In traditional network computer games, movement of the pointing device (e.g., the reticle) is not delayed—only the selection action is delayed (e.g., firing). This is because the local system can quickly render, say, a mouse cursor movement while the game action (e.g., clicking a button) needs to go back and forth to the authoritative server before rendering. This is in contrast to Puck Hunt which delays both the pointing and selection actions.

More appropriate is to compare Puck Hunt to user studies on cloud-based games where all user input is delayed by local, network, and remote processing. Two users studies [13] measured the impact of delay on cloud-based games using two different cloud game systems and two different third-person avatar games. One study used the game Crazy Taxi[14] on a commercial cloud-based game system, OnLive,[15] while the other study used the game Neverball[16] on an academic cloud-based game system, GamingAnywhere.[17] Both studies each had over 30 users play short game sessions with the users blind to controlled amounts of added delay. During each session, the system measured and recorded user in-game performance, and after each session users provided subjective opinions on the gameplay.

In order to compare the effects of delay across games, user objective performance results are normalized from 0 (worst) to 1 (best). For Puck Hunt, the models from Section 4.4 are similarly normalized and the added base delay (50 milliseconds) subtracted from the data since the cloud-based game measurements did not account for base delay. Figure 15 depicts the results. The x-axis is the added input delay and the y-axis is the user performance. There are three trendlines, one for each of the Puck Hunt models (WO-M, WPI-M, and WPI-T). The points are for the Crazy Taxi on OnLive and Neverball on GamingAnywhere, respectively. The model trendlines all accompany a similar space, suggesting they are indeed representing an inherent degradation in human performance despite the differences in target speeds and selection devices across the studies. The cloud-based games themselves are more varied, possibly owing to experimental noise, but they generally follow similar degradation paths as the models.

[14] http://en.wikipedia.org/wiki/Crazy_Taxi.
[15] http://onlive.com/.
[16] http://neverball.org.
[17] http://gaminganywhere.org/.

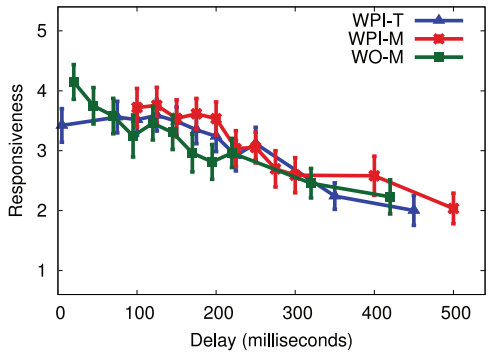

Fig. 16. Quality of experience versus delay.

4.9 Quality of Experience

While user opinion of delay often correlates with performance, subjective measures can ascertain the QoE beyond just the target selection time (or clicks). For Puck Hunt, for each delay and speed combination, users were asked to rate the responsiveness 1 (low) to 5 (high) (see Figure 4), with similar data gathered for the WPI-M and WO-M studies.

Figure 16 depicts a graph of the QoE—here, the responsiveness—versus delay for each model, using only data on targets with similar speeds (450 pixels/s for WPI-T and WPI-M and 550 pixels/s for WO-M). The x-axis is the total input delay and the y-axis is the reported responsiveness of the round. There are three trendlines, one for each user study. 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 perceive the delays. However, there is no noticeable separation of QoE for the different studies, suggesting the delay-based experience is similar regardless of the selection device (i.e., mouse or thumbstick).

5 DISCUSSION

The models presented in Section 4.3 may be of use to network game designers by illustrating a general relationship between target speed and delay and by providing a specific equation for selection time. For example, in a network game with target selection, the model in Equation (8) informs an interaction between delay and target speed when balancing the game. The model in Equation (9) may also be used to inform the use of delay compensation techniques. For example, when applying delay compensation techniques that prioritize actions based on delay sensitivity [14], the change in the time due to delay in Equation (9) is directly related to priority. Other latency compensation techniques may be similarly tuned or selected.

Hoffman [19] suggests target selection time increases *linearly* with delay (see Equation (7)). However, our observed curvature and model more likely corresponds to an exponential distribution (see Figure 7 and Table 3). The difference may be because Hoffmann's model is over a broader range of delays, from 30 to 1,000 milliseconds, and he notes that at the higher delay range (700+ milliseconds), users employ a "stop and wait" strategy in selecting the target, while a "continuous" strategy dominates lower delays. This may make a linear relationship the best fit for his data overall, but an exponential relationship a better fit for lower latencies, such as in our study.

Jagacinski [21] effectively suggests target selection time increases *linearly* with target speed (see Equation (4)). On the other hand, Hoffman [18] suggests target selection time increases *exponentially* with target speed (see Equation (5)). Based on Figure 8, they may both be right. For low delays

(200 milliseconds or below), the increase in selection time due to target speed appears linear, while for higher delays (300 milliseconds and above) the increase in selection time due to target speed appears exponential. Thus, low delay systems, such as delays solely from base delays, can model selection time with a linear component for target speed, while high delay systems, such as Internet games, would better model selection time with an exponential component for target speed.

Brady [6] analyzed the subjective experience of users selecting moving targets with a delayed mouse, finding a strong inverse linear relationship over the range of delays studied (from 0 to about 200 milliseconds). Our results (Figure 16) confirm Brady's, showing a significant inverse linear relationship (correlation −0.92) between the target selection time and the delay over a somewhat broader range (from 50 to 450 milliseconds).

Our final models fit with specific constants as presented (the equations in Table 6) likely hold primarily for the size of the target tested (100 pixels in diameter) and screen resolution (1,920×1,080 pixels), since target selection time is known to be affected by both target size and target distance (see Equation (2)). While the distance from the source (the starting location of the red ball/cursor) to the target varies in our study, the distance the ball travels averages about $\frac{1}{2}$ the maximum distance on the screen. Combining Fitts' index of difficulty (Equation (2)) with our results may produce a unified general model. Such modeling should consider both the absolute target size in pixels and also the target size relative to the screen resolution.

An additional consideration for specific models for target selection with delay is the skill of the user. Figure 11 shows high skill users are less impacted by delays from 250 to 450 milliseconds, with high skill users having only a 2.5-fold increase in selection time while low skill users have a 4-fold increase in selection time. Thus, for many practical delay ranges, the effects of skill may dominate the effects of delay. In fact, high skill users select targets faster with 350 milliseconds of delay than low skill users with 200 milliseconds of delay.

6 CONCLUSION

The growth in cloud technologies provides an opportunity for interactive cloud-based video services, such as computer games. However, the delay-sensitivity of computer games demands that the effects of delay on players be better understood in order to design cloud systems and software that support them. Previous studies measuring the effects of delay using commercial games do not generalize nor model well, and previous user input studies do not yet provide models unifying the effects of delay on player-game actions. In particular, the effects of delay on moving target selection, a common player action, using a game controller thumbstick, a common game input device, have not been explored.

Our work provides another step toward understanding the effects of delay on user input for games. We present the results of a user study with a custom game wherein users selected targets moving with different speeds using a thumbstick with delayed input. Over 50 users provided data for delays from 50 to 450 milliseconds and three target speeds—in total, over 8,000 observations of user performance and over 1,500 subjective quality assessments for the different delay and speed combinations.

Analysis of the results shows a slight increase in the time required to select a moving target for low delays (under 200 milliseconds), and a sharp increase in selection time for higher delays (over 250 milliseconds), especially for fast targets (450 pixels per second). Higher skilled users are less affected by delays than lower skilled users. Comparing the results with commercial games (UT 2003) and game genres (first person, third person, and omnipresent) shows the observed performance trends with delay match commercial first-person games well. Comparing the results to previous user studies with full-featured cloud games (Neverball and Crazy Taxi) on both commercial

(OnLive) and academic (GamingAnywhere) systems shows similar degradations in user performance with delay. Subjective opinions show users are perceptive of even modest delays.

A derived analytic model provides a good fit for the mean time to select a moving target, with an exponential term for delay and an important interaction term that captures the effects of target speed combined with delay. Comparing this model with two models from previous studies that used a delayed mouse instead of a thumbstick shows strong agreement in their accuracy, suggesting a basis for producing a scientific law that explains the time for a user to select a moving object with a pointing device when the input is subject to delay.

The results presented are relevant to system delays (e.g., local computer only) as well as network delays. In particular, the results pertain to cloud systems where all user input is sent to the cloud for rendering, meaning controller movements and clicks are delayed by the local system, network, and server.

However, traditional network games—where controller movement is processed and rendered by the local client—have only local delay for cursor/mouse movement, but incur additional delays for clicking since the latter have network and server processing delays, too. Modeling the effects of delays in such situations may be a fruitful area of future work.

In first-person games, target acquisition is often done by rotating the virtual world to pan the target to the reticle in the center of the screen. Future work studying whether the results in this article hold for first-person games, as suggested by Looser et al. [24], may be a useful confirmation.

The steering law [1] is a predictive model of human movement that describes the time required to navigate, or steer, through a two-dimensional tunnel derived from Fitts' Law. Since another common game action is steering—navigating an avatar/vehicle down a virtual pathway—future work might extend the steering law to incorporate delay (and perhaps moving paths).

Since clicking and missing a target has a cost in many games (i.e., using up virtual bullets), tasks that force users to consider number of clicks in addition to time may yield different models. In addition, there are many kinds of target movement in games, such as AI- or human-controlled so models that consider a richer set of avatar behaviors, not just speed, may be warranted.

Last, but not least, the 50 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 Robyn Domanico and Linh Hoang for conducting the thumbstick user studies and measuring base system delays for the experiments in this article, Marco Duran and Matthew Thompson for conducting the earlier mouse user studies, and to Ragnhild Eg and Kjetil Raeen for collaboration on the initial approach.

REFERENCES

[1] Johnny Accot and Shumin Zhai. 1997. Beyond Fitts' law: Models for trajectory-based HCI tasks. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*. 295–302.

[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*. 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'03)*.

[4] Tom Beigbeder, Rory Coughlan, Corey Lusher, John Plunkett, Emmanuel Agu, and Mark Claypool. 2004. The effects of loss and latency on user performance in Unreal Tournament 2003. In *Proceedings of ACM Network and System Support for Games Workshop (NetGames'03)*.

[5] Yahn W. Bernier. 2001. Latency compensating methods in client/server in-game protocol design and optimization. In *Proceedings of the Game Developers Conference*. [Online] http://www.gdconf.com/archives/2001/bernier.doc.

[6] Kyle J. Brady. 2015. *Taking Fitts Slow: The Effects of Delayed Visual Feedback on Human Motor Performance and User Experience*. Master's thesis. Arizona State University.
[7] K. Chen, P. Haung, G. Wang, C. Huang, and C. Lee. 2006. On the sensitivity of online game playing time to network QoS. In *Proceedings of IEEE Infocom*.
[8] Kuan-Ta Chen, Yu-Chun Chang, Hwai-Jung Hsu, De-Yu Chen, Chun-Ying Huang, and Cheng-Hsin Hsu. 2014. On the quality of service of cloud gaming systems. *IEEE Transactions on Multimedia* 26, 2 (Feb. 2014).
[9] Mark Claypool. 2005. 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.
[10] Mark Claypool and Kajal Claypool. 2006. Latency and player actions in online games. *Commun ications of the ACM* 49, 11 (Nov. 2006).
[11] Mark Claypool, Ragnhild Eg, and Kjetil Raaen. 2016. The effects of delay on game actions: Moving target selection with a mouse. In *Proceedings of the ACM Annual Symposium on Computer-Human Interaction in Play (CHI PLAY)*. Extended Abstract.
[12] 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'17)*.
[13] Mark Claypool and David Finkel. 2014. The effects of latency on player performance in cloud-based games. In *Proceedings of the 13th ACM Network and System Support for Games (NetGames'14)*.
[14] Mark Claypool, Tianhe Wang, and McIntyre Watts. 2015. A taxonomy for player actions with latency in network games. In *Proceedings of the 25th ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'15)*.
[15] Paul M. Fitts. 1954. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology* 47, 6 (June 1954), 381–391.
[16] 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 the 4th ACM Network and System Support for Games (NetGames'05)*.
[17] Abir Al Hajri, Sidney Fels, Gregor Miller, and Michael Ilich. 2011. Moving target selection in 2D graphical user interfaces. In *Proceedings of IFIP TC Human-Computer Interaction (INTERACT'11)*.
[18] Errol Hoffmann. 1991. Capture of moving targets: A modification of Fitts' law. *Ergonomics* 34, 2 (1991), 211–220.
[19] Errol Hoffmann. 1992. Fitts' law with transmission delay. *Ergonomics* 35, 1 (1992), 37–48.
[20] Zenja Ivkovic, Ian Stavness, Carl Gutwin, and Steven Sutcliffe. 2015. Quantifying and mitigating the negative effects of local latencies on aiming in 3D shooter games. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*. 135–144.
[21] R. Jagacinski, D. Repperger, S. Ward, and M. Moran. 1980. A test of Fitts' law with moving targets. *The Journal of Human Factors and Ergonomics Society* 22, 2 (April 1980), 225–233.
[22] Ricardo Jota, Albert Ng, Paul Dietz, and Daniel Wigdor. 2013. How fast is fast enough?: A study of the effects of latency in direct-touch pointing tasks. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*.
[23] R. Kerr. 1973. Movement time in an underwater environment. *Journal of Motor Behavior* 5, 3 (1973), 175–178.
[24] 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), 33–36.
[25] I. S. MacKenzie and W. Buxton. 1992. Extending Fitts' law to two-dimensional tasks. In *Proceedings of the ACM CHI Conference on Human Factors in Computing Systems*. 219–226.
[26] I. Scott MacKenzie and Colin Ware. 1993. Lag as a determinant of human performance in interactive systems. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. 6.
[27] Atsuo Murata and Hirokazu Iwase. 2001. Extending Fitts' law to a three-dimensional pointing task. *Elsevier Human Movement Science* 20 (2001), 719–805.
[28] James Nichols and Mark Claypool. 2004. The effects of latency on online Madden NFL Football. In *Proceedings of the 14th ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'04)*.
[29] 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'02)*.
[30] K. Raaen and R. Eg. 2015. Instantaneous human-computer interactions: Button causes and screen effects. In *Proceedings of the 17th HCI International Conference*.
[31] Kjetil Raaen and Andreas Petlund. 2015. How much delay is there really in current games? *Proceedings of ACM Multimedia Systems*.
[32] R. Shea, L. Jiangchuan, E. Ngai, and Y. Cui. 2013. Cloud gaming: Architecture and performance. *IEEE Network* 27, 4 (July-Aug. 2013), 16–21.

[33] R. So and G. Chung. 2002. Combined and interacting effects of hand and head movement delays on discrete manual performance in a virtual environment. *Ergonomics* 45, 2 (2002), 105–123.
[34] R. W. Soukoreff and I. S. MacKenzie. 2004. 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] Colin Ware and Ravin Balakrishnan. 1994. Target acquisition in fish tank VR: The effects of lag and frame rate. In *Proceedings of the Candian Information Processing Society's Graphics Interface Conference.*
[36] C. Ware and H. Mikaelian. 1987. An evaluation of an eye tracker as a device for computer input. In *Proceedings of ACM Human Factors in Computing Systems and Graphics Interface.*
[37] A. T. Welford. 1968. *Fundamentals of Skill.* Methuen, London, U.K.

Received August 2017; revised November 2017; accepted January 2018