On Models for Game Input with Delay – Moving Target Selection with a Mouse

Mark Claypool
Computer Science and Interactive Media & Game Development
Worcester Polytechnic Institute
Worcester, MA, 01609, USA
Email: claypool@cs.wpi.edu

***Abstract*—Networks and local systems add delays to user actions in computer games, increasing the time between user input and rendering on the screen. Top-down studies using games have helped understand the impact of delays, but often do not generalize nor lend themselves to analytic modeling. Bottom-up studies isolating user input can better generalize and be used in models, but have yet to be applied to computer games. Our work builds a custom game for studying delay and the fundamental user input of selecting a moving target with a mouse. Analysis of data from a large user study shows target selection time is exponential with delay, and provides for an analytic model based on delay and the interaction between delay and target speed.**

I. Introduction

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 with a mouse 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 [1], including system-level treatments (e.g., real-time priorities), latency compensation algorithms (e.g., dead reckoning) and even game designs to mitigate delay (e.g., delayed avatar response), an understanding of how latency affects fundamental player actions in games is needed in order to choose the most effective delay compensation 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 have tried to generalize to game genres (e.g., first person shooters) [2]–[7], but game design and game engines may obfuscate important system details, making it difficult to pertain to other games and to analytic modeling.

An alternate approach is bottom-up, studying fundamental user input differentiated by actions (e.g., target selection) or hardware (e.g., mouse, joystick). Contributions 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 predict the effects of delay for a wide-range of games and delay conditions.

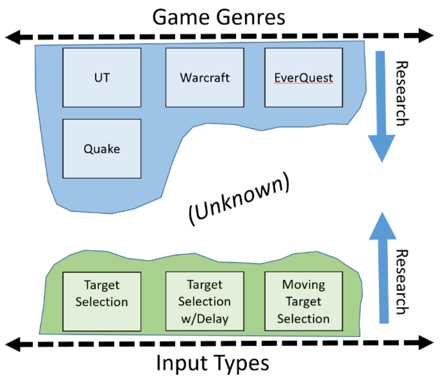

Fig. 1: Research in games and delay. Two approaches – top-down from existing games and bottom-up from user input.

While foundational studies of user input [8]–[10] have shown promise in modeling user interaction for computer systems, including games [11], such studies have not focused on game actions (e.g., moving target selection with a mouse) nor have they focused on delay ranges present in networked games. Ideally, game designers and system developers would have a model as far reaching and robust as Fitts' Law [8] – 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 delay. Our work takes a step towards providing such a model.

We design and implement a game that isolates the fundamental action of selecting a moving target with a mouse 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 provided by the user. We deploy the game in a user study with over 30 participants, with added 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 the mouse increases exponentially with delay. The time to select the target does not vary with target speed for low delays, but there are significant interaction effects between added delays and target speeds for high delays. User opinions on the quality of experience (responsiveness) show a pronounced linear decrease even for modest delay increases.

Lastly, we derive an accurate analytic model for the average time to select a moving target with a mouse based on delay and target speed. The model is exponential with delay and includes an interaction term for delay and target speed.

The rest of this paper is organized as follows: Section II describes work on user input modeling related to our work; Section III describes our methodology, including developing our game and conducting a user study; Section IV analyzes the user study results for overall trends and presents our model; Section V relates our results to earlier work and other systems; and Section VI summarizes our conclusions and outlines possible future work.

II. Related Work

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

A. Fitts' Law

Paul Fitts pioneered early seminal work in the area of human-computer interaction and ergonomics in the form of creating *Fitts' Law* [8]. With some simplification,[1] Fitts' Law describes 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):

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

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 and the constant k is determined empirically.

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 [14]) and input devices (e.g., eye tracking [15]).

B. 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 [10], 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.

[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.

Since many modern uses of Fitts' Law are for computer devices with two dimensional displays, MacKenzie and Buxton's [17] investigation of Fitts' Law provided guidelines for use of the law in evaluating pointing devices.

C. Fitts' Law with Moving Targets

Fitts' Law also only applies to stationary targets. This is entirely appropriate when applied to, say, a PC where a user is selecting a button with the mouse. However, this is less appropriate for a dynamic interface, such as a computer game, where the target is moving on the screen.

Jagacinski et al. [18] extended Fitts' Law with a revised index of difficulty (I, see Equation 2) that explicitly incorporates 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.

Hoffmann [19] refined Jagacinski's model with:

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

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 lost some intuition and simplicity when compared to Jagacinski's model.

Hajri et al. [9] refined Hoffmann's model by separating the index of difficulty 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 $\pm$ generalizes from $+$ in Hoffmann's model since the target may be moving towards (i.e., $-$) or away from (i.e., $+$) the source. A user study showed Hajri's model fit the experimental data.

D. Fitts' Law with Transmission 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, there are not only delays from the local computer system but also from network transmissions and processing on a remote computer system. This is particularly true in many network games where an authoritative server is responsible for processing all actions before rendering on the local client screen.

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

Suitable for this environment, Hoffman [12] 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 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 have move-and-wait input. Hoffman's experiments found the delay inflection point around 700 milliseconds.

Brady [13] experimented with target selection and delay, considering index of difficulty 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).

III. Methodology

To model user input and delay, we: 1) developed a game (*Puck Hunt*) that enables study of user input with controlled delay (Section III-A); 2) conducted a user study to evaluate the impact of a single user action with delay (Section III-B), and 3) analyzed the results of the user study through graphs and a model (Section IV).

A. Game

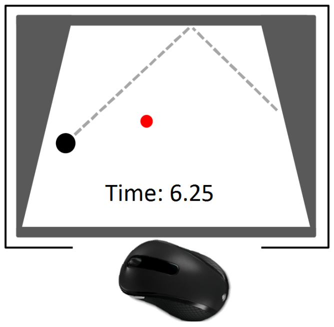

Fig. 2: Puck Hunt. Users click on moving target (puck) with mouse cursor (red ball). Game adds delay to mouse input and varies target speeds between each round.

We designed and developed a custom game called *Puck Hunt*[3] that allows for the study of a single user action with controlled amounts of delay. In Puck Hunt, depicted in Figure 2, 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. The user moves the mouse 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 clicking the mouse button. 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 any key, a new round starts, with the target at a new starting location with a new orientation and 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.

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 mouse input delays.

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

Puck Hunt is written in C++ using OpenGL with support from the Angel 2d game engine[4] to minimize the latency inherent in the software. Puck Hunt runs in fullscreen mode at 1080p resolution (1920x1080 pixels). The target is 100 pixels in diameter and the mouse cursor (the red ball) is 25 pixels in diameter.

Each round, the target moves with one of three possible speeds in Table Ia. Effectively, these speeds create levels of difficulty. The game adds a controlled amount of delay selected from the set in Table Ib. The delay is added to all mouse movements and button clicks 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 & speed combination appears 5 times, but the entire set of combinations is shuffled into a random order.

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

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.

B. User Study

Our user study was conducted in a windowless computer lab with bright, fluorescent lighting. The computers were Dell PCs with Intel i7-4790 4 GHz processors, 4 GB GeForce GTX 960 graphics cards and 16 GB of RAM, running Microsoft Windows 7. The monitors were 24" Dell U2412M LCDs with

[3]The name is a pun on the classic game *Duck Hunt* (Nintendo, 1984).

[4]http://angel2d.com/

a native resolution of 1920x1200 pixels and a refresh rate of 59p Hz.

Participants were volunteers solicited through WPI email lists, incentivized with a raffle for a $25 gift card.

First, users heard a scripted brief about the study and signed an Institute Review Board (IRB) consent form. 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 shift the mouse to whichever hand they preferred.

Users then completed a survey about demographics and gaming experience followed by launching the game.

Play commenced immediately, but the first two rounds were used for practice only and the results were not recorded. Play then proceeded through all 5x shuffled combinations of delay & speed (Table Ia and Table Ib), with one QoE question for each delay & 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.

Note, the delays in Table Ib 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], we measured the base delay for mouse actions on our lab computers using a Blur-busters type technique.[5]

A bread board with an LED was connected via a wire soldered to a mouse so that the LED lit up when the button was clicked. A high frame rate camera (a Casio EX-ZR200) filmed the player clicking on the QoE prompt, recording the action at 1000 f/s. By manually examining the individual video frames, the frame number when the light appears with the button click is subtracted from the frame number when the QoE prompt shows the input, giving the base delay.

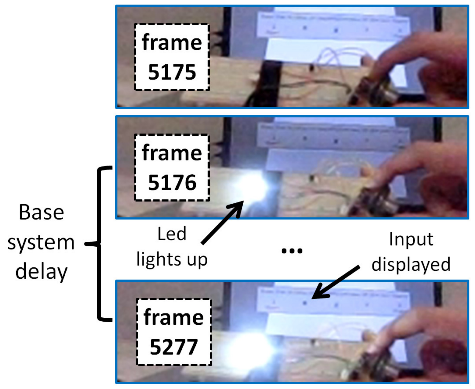

Fig. 3: Measuring base delay.

Figure 3 depicts the measurement method. The mouse is poised over the QoE prompt in frame 5175. In frame 5176, the button has been pressed indicated by the lit LED on the breadboard. The input is not displayed on the QoE prompt until frame 5277. Since there is one video frame each millisecond, subtracting 5716 from 5277 gives a base delay of 101 milliseconds.

The measurement method was repeated 5 times, resulting in base delay values of 93, 99, 101, 101 and 112 milliseconds. Hence, 100 milliseconds is added to all delay analysis.

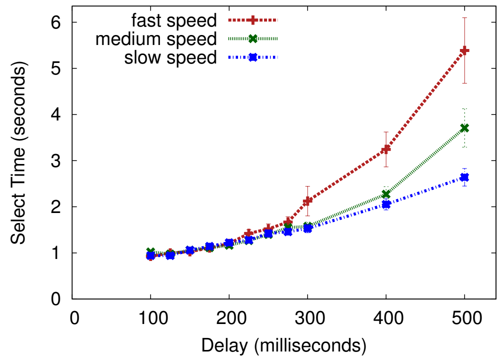

Fig. 4: Selection time versus delay, grouped by target speed.

IV. Analysis

Thirty-two users participated in the study. Ages ranged from 18-26 years with a mean and median of 21. Twenty-three identified as male, 8 as female, and 1 did not specify. The mean self-rating as a PC gamer (scale 1–5) was 3.6, showing a slight skew to having “high ability”. Exactly half the users played 6+ hours of computer games per week, about the same fraction that used a computer (PC/Mac) with a mouse 6+ hours/week.

A. *Selection Time - Measurement*

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

Figure 4 depicts selection time versus delay, analyzed 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 trend-lines, one for each target speed tested. Each point is the mean time for all users for that delay & 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 impact mean selection time. However, starting at delays of 225 milliseconds (for fast targets) and 400 milliseconds (for medium speed target), the faster speed targets become harder to select than the slowest speed targets. At the extreme delay (500 milliseconds), the fast targets take 5x longer to select than when there is minimal delay (100 milliseconds) and even the slow targets take over 2.5x longer to select.

Figure 5 depicts selection time versus speed, analyzed by delay. The x-axis is target speed in pixels per second and the y-axis is the time to select the target. There are five trend-lines, one for each of the total input delays (added delay + base delay).[6] Each point is the mean time for all users for that

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

[6]The other delays tested are not shown to keep the graph readable.

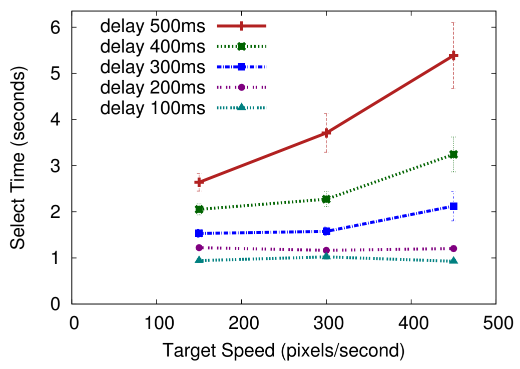

Fig. 5: Selection time versus speed, grouped by delay.

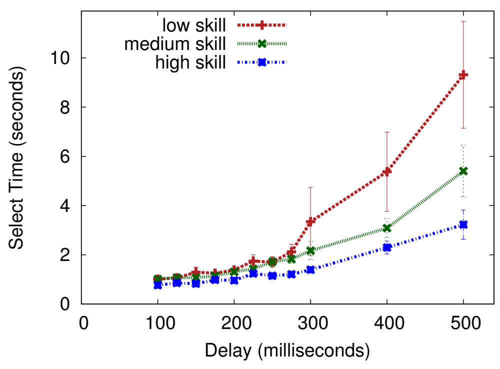

Fig. 6: Selection time versus delay, grouped by user skill.

delay & 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. As seen in alternate form in Figure 5, for delays of 200 milliseconds and under, the lines are flat – the speed of the target does not impact mean target selection time.

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 spend more time on the computer, in general, and more time playing computer games where moving target selection is common, especially, are likely more skilled in this task.

For our study, users provided a self-rating of PC gamer skill, from 1 (low) to 5 (high). Based on our user sample, we triaged users into low skill (6 users with rating 1-2), medium skill (15 users with rating 3-4) and high skill (12 users with rating 5).

Figure 6 depicts selection time versus delay for the fast targets only, analyzed by self-reported skill. The axes are as for Figure 4. There are three trend-lines, one for each skill group. Each point is the mean time for all users with that skill for fast speed targets for that delay, shown with a 95% confidence interval. Overall, the increase in mean selection time as delay increases holds for all skill groups. For delays under about 300 milliseconds, all skill groups perform comparably. However, there is clear separation of the skill trend lines for high delays, with the most skilled group being affected the least and the least skilled group the most. At the extreme delay (500 milliseconds), the least skilled users take about 3x longer to select the target than the most skilled users.

B. Selection Time - Model

While analysis of trends for moving target selection times with delay provides valuable insights for game researchers and developers, analytic models are more flexible at representing the relationships. With this goal in mind, we modeled the mean time to select a moving target with delayed mouse actions.

TABLE II: User study data values used for standardization.

Factor	Mean	StDev
Delay	245 milliseconds	114
Speed	300 pixels/second	122

The previous analysis shows users' mean selection times trend upward with increased delay. The observed curvature corresponds to possibly a linear, but more likely an exponential distribution. The selection times trend with target speed is less clear – for the range of speeds tested, target speed has a negative impact on selection time for large delays, but almost no effect for small delays. This points to important interactions between speed and delay to incorporate into a model.

Thus, we propose modeling the time to select a moving target with a mouse (T) with exponential terms 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 determined empirically through user study. We standardized our user study data[7] using the values in Table II.

Fitting a regression model to the standardized data[8] yields an adjusted R^2 of 0.97, F-stat 328 and $p < 2.2 \times 10^{-16}$. The simplified final model for the time to select a moving target with a delayed mouse (T) is:

$$T = 1 + 0.2e^d - 0.04e^s + 0.1e^d e^s \tag{9}$$

where d and s are the standardized delay and speed, respectively: $d = \frac{D-245}{114}$ and $s = \frac{S-300}{122}$. All terms are significant at $p < 0.001$, except the main effect of speed ($p < 0.2$).

C. Mouse Clicks - Measurement

Another problem reported by some Fitts-type user studies is that an "aggressive" user, clicking 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., virtual bullets). In our case, all clicks above one per target are misses.

[7] Subtracting the mean and dividing by the standard deviation.
[8] Using R, https://www.r-project.org/

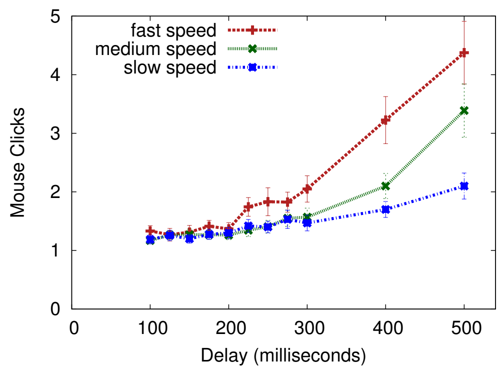

Fig. 7: Mouse clicks versus delay, grouped by target speed.

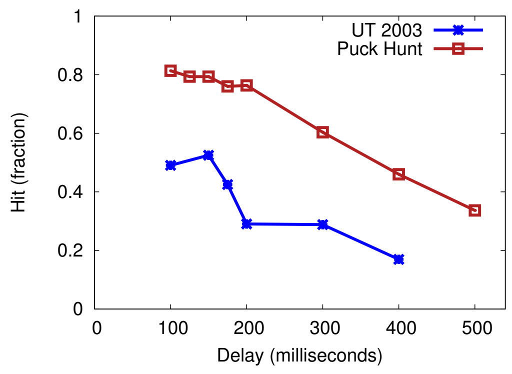

Fig. 8: Hit fraction versus delay, grouped by game.

Figure 7 depicts mouse clicks versus delay, analyzed by target speed. The x-axis is the total input delay and the y-axis is the number of mouse clicks needed to select the moving target. There are three trend-lines, one for each target speed tested. Each point is the mean number of mouse clicks for all users for that delay & speed combination, shown with a 95% confidence interval. Overall, there is an increase in mean mouse clicks as delay increases. Remarkably similar to selection time, clicks increase exponentially over the range of delays tested. For delays under 200 milliseconds, the speed of the target does not impact the number of clicks. However, starting at delays of 225 milliseconds (for fast targets) and 400 milliseconds (for medium speed target), the user misses more for faster speed targets than for the slowest targets. At the extreme delay (500 milliseconds), users miss about ⅔ to ⅘ of the time for slow and fast targets, respectively.

The performance of the users in Puck Hunt can be compared to users in commercial computer games. A previous study [7] 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. Figure 8 depicts the results. The x-axis is the total input delay and the y-axis is the hit fraction (i.e., for Puck Hunt, this is 1 divided by the number of mouse clicks). There are two trend-lines, one for UT 2003 and one for Puck Hunt, averaged across all target speeds. Overall, the two trend-lines follow the same pattern, decreasing approximately linearly with an increase in delay. The Puck Hunt hit fractions are significantly higher than the UT 2003 hit fractions, but this may be because the UT 2003 avatar was controlled by a human and moved unpredictably, while the Puck Hunt target moved with constant velocity.

D. Comparison

In order to better understand the impact of delay and target speed on the time to select a moving target with a mouse, it is helpful to compare the results to user studies from network games. Previous work has found the effects of latency depend upon user in-game perspective [6]. The game perspective defines 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. First person shooter (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 games (RTS) and construction and simulation games are examples of game genres with the omnipresent perspective.

In order to compare the effects of delay across games, user objective performance results are normalized from 0 (worst) to 1 (best). Results from previous studies[9] of latency and traditional network games (first person avatar [7], [21], third person avatar [22], [23], and omnipresent [5]) are similarly normalized and fit with an exponential curve [6]. The same is done for our user study data, normalizing the selection time and mouse clicks for the fastest targets.[10] In order to make our data comparable to the previously published results, the added base delay (100 milliseconds) is subtracted from our user study data.

Figure 9 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 latency tolerance threshold depends on the

[9]Providing details on these studies is not feasible given the space constraints, but the interested reader is encouraged to follow the references.

[10]Slow and medium targets were just above the fast targets but made the graph less readable.

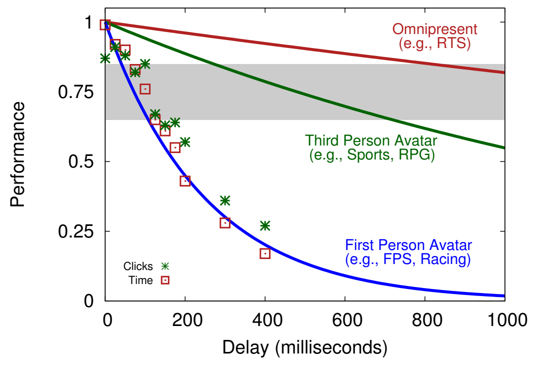

Fig. 9: User performance versus delay.

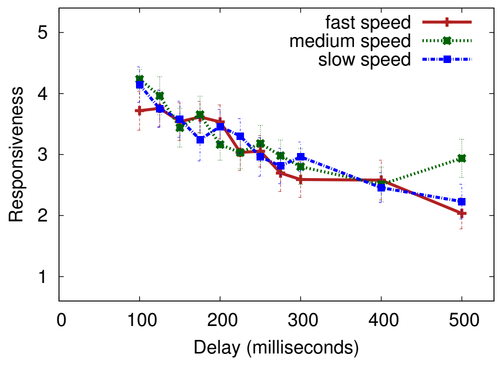

Fig. 10: QoE versus delay, grouped by target speed.

game and to some extent the users own perception and sense of immersion (hence the gray color and the rectangle shape rather than a line).

The time to select a moving target with a mouse and the number of mouse clicks most closely follows the first person avatar model of perspective. This is likely because performance in such games is most closely attuned to the exact actions of the user (e.g., aiming a weapon) and so is severely impacted by delay, whereas in other game models user performance is handled by agents with some autonomy so the effects of delay are mitigated. Note, this trend is similar to that of cloud-based games [24], where all user input is delayed by local, network and remote processing.

E. *Quality of Experience*

While user opinion of delay often correlates with performance, subjective measures can ascertain the quality of the experience (QoE) beyond just the target selection time (or mouse clicks). For Puck Hunt, for each delay & speed combination, users were asked to rate the responsiveness 1 (low) to 5 (high).

Figure 10 depicts a graph of the quality of experience – here, the responsiveness – versus delay. The x-axis is the total input delay and the y-axis is the responsiveness of the round. There are three trend-lines, one for each target speed. Each point is the mean rating for all users for that delay & speed combination, 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, unlike for performance, there is no noticeable separation of QoE with target speed suggesting users gauge responsiveness based on delay independently of the difficulty of the action.

V. Discussion

Hoffman [12] 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 4 and Equation 9). The difference may be because Hoffmann's model is over a broader range of delays, from 30 to 1000 milliseconds, and he notes that at the higher range (700+ milliseconds), users employ a "stop and wait" strategy in selecting the target, while a "continuous" strategy dominates lower latencies. This may make a linear relationship the best fit for his data overall, but an exponential a better fit for lower latencies, such as in our study.

Jagacinski [18] effectively suggests target selection time increases *linearly* with target speed (see Equation 4). On the other hand, Hoffman [19] suggests target selection time increases *exponentially* with target speed (see Equation 5). Based on Figure 5, they may both be right. For low delays (200 milliseconds or below), the increase in selection time due to target speed is linear, while for higher delays (300 milliseconds and above) the increase in selection time due to target speed is 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 [13] 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 10) confirm Brady's, showing an inverse linear relationship (correlation -0.88) between the target selection time and the delay over a somewhat broader range (from 100 to 500 milliseconds).

Our final model as presented (Equation 9) likely holds primarily for the size of the target tested (100 pixels in diameter) and screen resolution (1920x1080 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 mouse pointer) to the target varies in our study, the distance the mouse travels averages about ½ 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 concern of any unified model for target selection with a mouse is the skill of the user. Figure 6 shows high skill users are less impacted by delays from 300 to

500 milliseconds, with high skill users having only a 3-fold increase in selection time while low skill users have a 10-fold increase in selection time. Thus, for many practical delay ranges, the effects of skill may dominate the effects of delay.

The results presented are relevant to all forms of input delay, both the local system (e.g., operating system and hardware) and the network. In particular, the results pertain to cloud systems where all user input is sent to the cloud for rendering, meaning mouse movements and clicks are delayed by the local system, network and server. However, traditional network games – where mouse movement is processed and rendered by the local client – have only local delay for mouse movement, but incur additional delays for mouse clicking since the latter have network and server processing delays, too.

VI. Conclusion

Understanding the effects of delay can help game designers and researchers develop and deploy solutions to mitigate the negative impact of delay on game players. Previous game studies measuring the effects of delay do not generalize nor model well, and previous user input studies do not yet provide models quantifying the effects of delay on player actions.

Our work provides a step towards understanding the effects of delay on user input for games. We present results of a user study with a custom game wherein players selected moving targets with different speeds using a mouse with delayed input. Over 30 users provided data for delays from 100 to 500 milliseconds and 3 target speeds – in total, over 5000 observations of user performance and over 1000 subjective quality assessments for the different delay & speed combinations.

Analysis of the results shows an increase in the time required to select a moving target even for low delays (under 200 milliseconds), and a sharp increase in selection time for higher delays (over 300 milliseconds) and fast targets (450 pixels per second). Subjective opinions show users are sensitive to 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.

Hoffman [12] found a delay inflection point around 700 milliseconds above which users changed behavior from continuous movement to move-and-wait. While such delays are typically beyond what most game players tolerate, future work could explore this inflection point for other applications. Since user skill significantly impacts target selection time (see Figure 6), future work may look to first quantify skill and then incorporate it into an analytic model. Additional models can be derived for mouse clicks and quality of experience. Other forms of user input for target selection (e.g., analog controller, touch on mobile/tablet) or user input in the form of keyboard or game controller buttons could also be explored.

Acknowledgments

Thanks to Marco Duran and Matthew Thompson for conducting the user studies and measuring base system delays, and to Ragnhild Eg and Kjetil Raeen for collaboration on the approach.

References

[1] Y. W. Bernier, "Latency Compensating Methods in Client/Server In-game Protocol Design and Optimization," in *Proceedings of the GDC*, San Francisco, CA, USA, Feb. 2001.

[2] R. Amin, F. Jackson, J. E. Gilbert, J. Martin, and T. Shaw, "Assessing the Impact of Latency and Jitter on the Perceived Quality of Call of Duty Modern Warfare 2," in *Proceedings of HCI – Users and Contexts of Use*, Las Vegas, NV, USA, Jul. 2013, pp. 97–106.

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

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

[5] M. Claypool, "The Effect of Latency on User Performance in Real-Time Strategy Games," *Elsevier Computer Networks*, vol. 49, no. 1, pp. 52–70, Sep. 2005.

[6] M. Claypool and K. Claypool, "Latency and Player Actions in Online Games," *Communications of the ACM*, vol. 49, no. 11, Nov. 2006.

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

[8] P. M. Fitts, "The Information Capacity of the Human Motor System in Controlling the Amplitude of Movement," *Journal of Experimental Psychology*, vol. 47, no. 6, pp. 381–391, Jun. 1954.

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

[10] I. S. MacKenzie and W. Buxton, "Extending Fitts' Law to Two-Dimensional Tasks," in *Proceedings of the ACM CHI Conference on Human Factors in Computing Systems*, May 1992, pp. 219 – 226.

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

[12] E. Hoffmann, "Fitts' Law with Transmission Delay," *Ergonomics*, vol. 35, no. 1, pp. 37 – 48, 1992.

[13] K. J. Brady, "Taking Fitts Slow: The Effects of Delayed Visual Feedback on Human Motor Performance and User Experience," Master's thesis, Arizona State University, Dec. 2015.

[14] R. Kerr, "Movement Time in an Underwater Environment," *Journal of Motor Behavior*, vol. 5, no. 3, pp. 175 – 178, 1973.

[15] C. Ware and H. Mikaelian, "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.

[16] A. T. Welford, *Fundamentals of Skill.* London, U.K.: Methuen, 1968.

[17] R. W. Soukoreff and I. S. MacKenzie, "Towards a Standard for Pointing Device Evaluation – Perspectives on 27 Years of Fitts' Law Research in HCI," *Elsevier International Journal of Human-Computer Studies*, vol. 61, no. 6, pp. 751 – 789, 2004.

[18] R. Jagacinski, D. Repperger, S. Ward, and M. Moran, "A Test of Fitts' Law with Moving Targets," *The Journal of Human Factors and Ergonomics Society*, vol. 22, no. 2, pp. 225–233, Apr. 1980.

[19] E. Hoffmann, "Capture of Moving Targets: A Modification of Fitts' Law," *Ergonomics*, vol. 34, no. 2, pp. 211–220, 1991.

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

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

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

[23] J. Nichols and M. Claypool, "The Effects of Latency on Online Madden NFL Football," in *Proceedings of ACM NOSSDAV*, Kinsale, County Cork, Ireland, Jun. 2004.

[24] M. Claypool and D. Finkel, "The Effects of Latency on Player Performance in Cloud-based Games," in *Proceedings of the 13th ACM Network and System Support for Games (NetGames)*, Nagoya, Japan, Dec. 2014.