Effects of Local Latency on Game Pointing Devices and Game Pointing Tasks

Michael Long
University of Saskatchewan
Saskatoon, Canada
michael.long@usask.ca

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

ABSTRACT

Studies have shown certain game tasks such as targeting to be negatively and significantly affected by latencies as low as 41ms. Therefore it is important to understand the relationship between local latency – delays between an input action and resulting change in the display – and common gaming tasks such as targeting and tracking. In addition, games now use a variety of input devices, including touchscreens, mice, tablets and controllers. These devices provide very different combinations of direct/indirect input, absolute/relative movement, and position/rate control, and are likely to be affected by latency in different ways. We performed a study evaluating and comparing the effects of latency across four devices (touchscreen, mouse, controller and drawing tablet) on targeting and interception tasks. We analyze both throughput and path characteristics, identify differences between devices, and provide design considerations for game designers.

CCS CONCEPTS

• **Human-centered computing → Pointing devices**; • **Applied computing → Computer games**;

KEYWORDS

Lag; Latency; Game experience; Pointing devices.

ACM Reference Format:
Michael Long and Carl Gutwin. 2019. Effects of Local Latency on Game Pointing Devices and Game Pointing Tasks. In *CHI Conference on Human Factors in Computing Systems Proceedings (CHI 2019), May 4–9, 2019, Glasgow, Scotland Uk.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3290605.3300438

1 INTRODUCTION

Lag is a problem for many computing tasks, but is particularly disruptive in games [7]. Fast pacing and time pressure make games far more sensitive to latency than other applications. One type of lag is local latency – the delay between an input action and resulting output. Although special hardware such as 120Hz monitors can reduce the problem, local latency up to 250ms is still common in real-world setups [18].

Local latency affects many core game interaction mechanics, such as targeting, tracking, and target interception. In this paper we focus on two task types: static target acquisition and moving target acquisition. Static target acquisition (targeting) is extremely common in many games. Examples include selecting towers in tower defence games, unmoving unit selection in real-time strategy games, rhythm games like Osu!, and objects in "clicker" or Candy Crush-style games. Targeting is very sensitive to latency, having been shown to be affected by latencies as low as 41ms [18]. Moving target acquisition (interception) is a related task to stationary targeting, and is also difficult with high amounts of latency [6]. Selecting units in RTS games, shooting in Duck Hunt, and hitting falling objects in Fruit Ninja all feature interception.

Targeting and interception performance is highly dependent on the type of pointing device used. Games are played on various gaming platforms, thus using a variety of pointing devices. Common devices include mice (PC games), touchscreens (mobile games) and gamepads (console games). Pointing devices can be categorized using several fundamental dimensions, such as direct versus indirect input, absolute versus relative movement, and positioning style (isotonic, elastic or isometric). These different characteristics suggest that devices may be affected differently by local latency. Previous work provides initial evidence: for example, users are extremely sensitive to latency on touchscreens, and can notice latencies as low as 2ms [29]. Jota et al. found dragging with direct-touch input to be significantly affected by latencies as low as 25ms [20]. The mouse has been extensively studied, and generally performs very well for pointing [3, 7, 8, 26]. Although less common, drawing tablets have also shown differences when used in either direct or indirect fashion. Deber found a large just-noticeable-difference between direct and indirect touch input when tapping and dragging [9].

Every step in the process of capturing input and displaying the result adds local latency, making it impossible to reduce latency to zero in a real-world game. Therefore, understanding the effects of latency on common tasks with common input devices is important for game designers and hardware manufacturers. Several studies have examined the effects of lag on individual devices, but broader studies that cover several devices and latencies are rare – and multi-device studies would be particularly useful for assessing effects on multiplatform games and games that support cross-platform play (e.g., PC and console multiplayer).

To provide game designers with more knowledge about input devices and their characteristics, we performed a study using four pointing devices. Input devices were chosen to represent common gaming devices whilst covering a broad range of characteristics. Chosen devices included touchscreen, drawing tablet, mouse and gamepad. The touchscreen is unique in not requiring continuous visual feedback with a cursor while still requiring synchronization between hand movement and tapping for interception tasks. The drawing tablet was included for its unique input characteristics and its use by top players of popular rhythm games such as Osu!. Additionally, both PS4 and Steam controllers have touchpads which can function like a tablet. We examined these four devices by having participants perform stationary and moving target acquisition tasks under eight levels of latency. Our analysis included throughput whilst also examining cursor path characteristics such as movement variability and target re-entry that provide additional information over and above regular performance measures [25].

Our work provides game designers with new information about how different input devices will be affected by local latency for core game actions. Our results show that local latency has significant negative effects on performance, device throughput, and player experience – but that not all devices were equally affected by latency. For example, stationary targeting with a touchscreen was almost completely unaffected by latency, and the touchscreen was also the best device for target interception. The performance of the gamepad and the drawing tablet was very similar, despite the large underlying differences between these two devices. The mouse performed reasonably well (but below the touchscreen, surprisingly). Our path analyses help to explain some of these differences: two path metrics (target re-entry and movement error) were strongly correlated with both performance and latency. Our analyses show that the controller produces characteristic paths with large zig-zags, and that the drawing tablet produces erratic paths and is prone to slipping off the target when tapping. Lastly we provide several design suggestions such as using direct input in situations of high latency, using aim assist with controllers and drawing tablets, using wider corridors for movement with a controller, and using physical buttons rather than virtual buttons with drawing tablets.

2 RELATED WORK

2.1 Network vs. Local Latency

Network latency is the delay in the transmission of data over a network, and is well-studied (e.g., [3, 4, 7, 8, 33]). Many games include networked multiplayer modes, requiring player avatar locations to be synchronized across machines, leading to the problem of an avatar's true location differing from where they appear on different computers [6, 11]. Quax et al. demonstrated network latencies as low as 60ms significantly affect targeting in first-person shooters. Network latency has been shown to disrupt two-person coordination [12], but these effects can be reduced if delays are visualized [13] or the movement is predictable [30]. Networks can also cause jitter (variance in latency), which affects interpretation of smooth motion and streams [12].

Our study examines local latency, which does not involve any network delays. There are multiple factors that contribute to local latency [18]. The first is the polling rate of input devices such as mice or touch sensors. Regular USB mice poll at 125Hz or 250Hz (4-8ms), and typical gaming mice poll at 1000Hz (1ms). Second, game software typically operates in discrete frames, and can take multiple frames to process input. Third, monitors have refresh rates ranging from 30-144Hz (with television post-processing adding 30-60ms more), and pixels themselves can take 1-12ms to change colour. A survey of real-world gaming setups found overall local latencies ranging from 23 to 243ms [18].

2.2 Performance Metrics

Fitts' Law is well known in HCI research as a predictive model for pointing time [22] in which each task has an Index of Difficulty (ID) based on distance to and width of the target (we use the Shannon formulation: $ID = log_2(D/W + 1)$ [23]). Index of Performance (IP, or throughput) is the the main performance measure, and is adjusted based on ID: $IP = ID/MT$ (in bits/s). Fast performance results in a higher IP. Movement Time (MT) is task duration, and is guaranteed to increase with latency.

Several extensions to Fitts' Law have been proposed, including an extension to model moving targets [19], and additional terms for latency. Hoffman proposes a linear latency term [17], whilst Mackenzie and Ware propose a multiplicative latency term [27, 34]. Claypool goes further, and suggests it should be exponential [4, 5]).

Mackenzie identified other measures related to the path of travel during a pointing task [25]. These measures provide a richer description of pointing tasks than simply analyzing

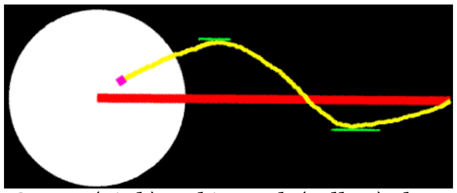
Figure 1: Cursor (pink) and its path (yellow) along task axis (red). This path features one task-axis crossing, and two movement direction changes (green).

throughput and movement time; they give multiple definitions of accuracy, showcasing the fundamental differences between pointing devices. Several of the metrics mention a task axis, which is the optimal straight-line path from the cursor's start position to the target center (Figure 1), and y is a sample point on actual cursor path.

Target Re-entry (TRE): Number of times the cursor entered the target region (excluding the first). A TRE of 1 means the cursor entered, left, then re-entered target.

Task Axis Crossing (TAC): Number of times the cursor crossed the task axis on way to target.

Movement Direction Change (MDC): Number of cursor path direction changes relative to task axis. Correlated with TAC, as a direction change may cause a TAC.

Orthogonal Direction Change (ODC): Number of path direction changes relative orthogonally to task axis.

Movement Offset (MO): Mean distance deviation of sample points from task axis (negative values indicate being below the task axis, and positive above). Formula: $MO = \bar{y}$.

Movement Error (ME): Average deviation of sample points from task axis, irrespective of whether points are above or below task axis. Formula: $(\sum |y_i|)/n$, where y_i is distance from task axis to a sample point.

Movement Variability (MV): Variability of distance between sample points and task axis (standard deviation of ME). Formula: $MV = \sqrt{(\sum(y_i - \bar{y})/(n-1)}$.

Misclicks (MC): Number of clicks outside the target.

2.3 Pointing Devices

Our study used four input devices (touchscreen, drawing tablet, thumbstick on gamepad, and mouse), which vary on several fundamental properties [24].

Direct vs. Indirect: whether visual output is displaced from input space (i.e., touchscreens share input and output space).

Absolute vs. Relative Movement: the mapping between points in input space and output space. The mapping is consistent for absolute movement (e.g., tapping the corner of a touchscreen always results in a corner tap), but is variable for relative movement (e.g., clutching).

Property Sensed: the type of physical change sensed by the device. Common sensors for pointing devices include position, displacement, and force.

Property Controlled: on-screen cursor property that is changed. Most devices control position (relative or absolute) or velocity.

In the following sections we provide a brief overview of each device and review results from prior work that are relevant to our investigation of local latency.

Touchscreen. Touchscreens are ubiquitous in mobile phones and tablets, and are increasingly popular as monitors for desktops and laptops. Touchscreens use direct input and absolute movement, and sense and control position. Touch latency is a key factor in user experience [2], with users perceiving as little as 2ms of delay when dragging with direct touch input [29]. Previous work has found various latency thresholds for when performance starts to degrade, including 25ms, 60ms, and 100ms [2, 15, 35]. Researchers have also considered methods for reducing touchscreen latency, by relaxing synchronization restrictions introduced to reduce screen tearing and framerate control; these techniques can help reduce touch latency, sometimes dramatically [35].

Drawing Tablet. Drawing or graphics tablets provide a stylus that is used to touch the surface, allowing a tripod-like grip to perform precise movements. Tablets provide indirect input and absolute movement, and sense and control position. One of the few performance studies of drawing tablets (involving mouse, trackball and tablet) showed mice and tablets had similar performance for dragging and pointing [26]. Tablets are of interest because some expert players of some rhythm games (such as Osu!) primarily use tablets because they allow absolute movement without occlusion of the display space.

Mouse. The mouse features indirect input and measures displacement for its relative cursor movement. Despite requiring users to learn a new mapping from input to output space, the mouse has become ubiquitous. Mice have been widely studied in performance analyses, with most Fitts Law adaptations having been verified using a mouse [4, 17, 27]. Several studies have specifically examined mouse targeting in videogames [3, 7, 8], with Claypool finding speed of interaction closely linked to player performance and latency in target interception [8]. Long et al. used a mouse to determine that target interception is significantly affected by latencies as low as 50ms, and also created a latency model to predict error rates in target interception [21].

Gamepad. Gamepads are primarily used with consoles, and normally include thumbsticks. Thumbsticks provide indirect input and relative movement, but unlike the mouse, they typically sense force and control the velocity of a screen object (e.g., a targeting reticle). They also have a limited range of motion, making the maximum velocity game-dependent. Gamepads and latency have been studied less often than mice, although Claypool examined the performance of thumbstick

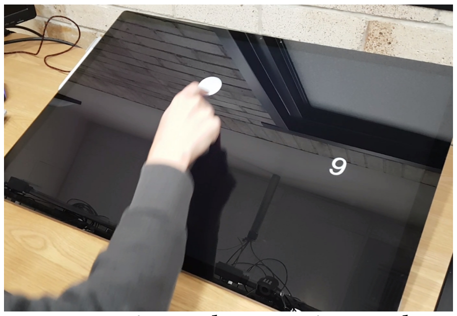
Figure 2: Experimental setup using touchscreen.

target interception and found latency to exponentially affect selection, especially above 250ms [5].

Multi-Device Studies. It is useful to compare and contrast multiple devices across a standard task, but there are relatively few of these studies – particularly involving latency. General comparisons of devices have been carried out by Mackenzie (mouse, trackball, joystick and touchpad [25]), and Claypool has considered latency in studies of the mouse and gamepad [4, 5]). We know of no single study that has compared a wide range of pointing devices under latency conditions.

3 EXPERIMENT METHODS

3.1 Participants

Sixteen participants (8 male, 7 female, 1 other, mean age 24) were recruited from a local university. All but one were right-handed, and all used the mouse in their right hand. Thirteen played videogames in a typical week, with twelve having previously experienced lag at some point. In a typical week, participants estimated an average of 29 hrs/week using a mouse, 22 hrs/week using a touchscreen, and 4 hrs/week with a controller, and only two participants regularly used a drawing tablet. Participants self-reported their proficiency with each device on a 1-5 scale (1 being none, 5 being expert): averages were 4.7 for mouse, 4.3 for touchscreen, 3.5 for controller, and 2.3 for drawing tablet.

3.2 Apparatus

The custom pointing task was built using Unity3D, and the software recorded all study data. The study ran on a Microsoft Surface Studio with a Core i7 2.7GHz CPU and Windows 10 Pro, an NVidia GTX 980m video card, 32Gb of RAM, and a 4500x3000 28" touch display. Input devices included an MSI Interceptor DS100 mouse (1000Hz, 800dpi, Windows Mouse Acceleration disabled), Wacom Intuos S drawing tablet (read rate of 133 Hz, Windows Ink disabled) and a wired USB Xbox gamepad. The touchscreen was the monitor of the Surface Studio itself. The game ran at a constant 120fps, with a 60Hz monitor.

Question	Source
Q1: I felt capable and effective when playing.	PENS: Competence [32]
Q2: Using the current input device was fun.	IMI: Enjoyment [28]
Q3: I put a lot of effort into those rounds.	IMI: Effort [28]
Q4: How well I did was completely due to me.	Attribution [10]
Q5: I was frustrated by the task.	TLX: Frustration [14]
Q6: The cursor movement was responsive.	Custom

Table 1: In-game experience questions asked after every second latency level.

Following procedures from prior work [18], base system latency was calculated using a 240fps camera (4.17 ms/frame) that recorded both mouse (1000Hz) and screen. Review of the frames from cursor movement to screen update showed an average of 52ms local latency (using 10 samples). The Xbox controller was found to have 8ms of latency using the testing method described above. The drawing tablet and the touchscreen have native latencies of 8ms, and the mouse polling rate was set to a matching 125Hz. Since all devices had 8ms of delay, there was a total of 60ms of local latency. All graphs, figures and text include this 60ms. Additional artificial latency (Table 2) was simulated at the input device level by buffering input for later use.

3.3 Procedure

Participants first completed informed consent and demographic questionnaires, and performed a reaction time test [1]. Participants then performed 864 rounds of target selection (including 96 practice rounds). Each round had a maximum time of 10 seconds.

Due to time constraints, participants completed an in-game player experience questionnaire only after every second level of latency (Table 1). Participants could rest after each round. Additional subjective player experience questions were asked at session end. On average the experiment took 45 minutes to complete.

3.4 Design

We built a custom game that emulated classic pointing and interception tasks. For each task, the participant moved from the center of the screen and clicked (or tapped) on a white target circle. We controlled target distance, size, and movement. Direction to the target was pre-randomized such that all participants had the same angles to the target.

The study was a within-participants design with five main factors: distance to target, target size, target speed, latency level and device used. Each round used one of three distances to target (600, 1200 and 1800 pixels – 79mm, 158mm and 237mm), one of two target widths (300 and 600 pixels – 39.5mm and 79mm), one of two target movement speeds (0 pixels/sec and 1200 pixels/sec – 0mm/s and 158mm/s), a level of added latency (0-350ms, Table 2), and an input device

Level	1 P	2	3 S	4	5 S	6	7 S	8	9 S
Added Latency (ms)	0	0	50	100	150	200	250	300	350
Total Latency (ms)	60	60	110	160	210	260	310	360	410

Table 2: Local latencies per round; P=practice. S=survey after round. Each level repeated 24 times per device.

(mouse, controller, touchscreen or drawing tablet). Device order was balanced using a Latin square.

Target position, target direction from the screen center, and target speed were repeated for each device/latency pairing. Each latency level consisted of 24 rounds (3 distances x 2 sizes x 2 speeds, repeated twice). Our three target distances and two target sizes give six indices of difficulty (ID): 1, 1.59, 2, 2.32, 2.81. Targets were stationary for the first 12 rounds of a latency level, and moving for the final 12 rounds. Latency was presented in increasing order, allowing the user to adapt to the latency – performance differences due to latency must therefore overcome the learning effect.

Cursor position was recorded in each frame and then combined to form a cursor path for analyzing path metrics. Consecutive duplicate stationary points were discarded to remove initial hesitation. The touchscreen had no continuous path to record and is not included in these measures. All formulas assume task axis is $y = 0$. To better view latency effects aside from the minimum added delay, we adjusted all movement time values by removing the latency amounts (Figure 3). Data was not recorded from practice rounds.

4 RESULTS: STATIONARY TARGETS

4.1 Movement Time

As seen in Figure 3, the touchscreen (the only direct-pointing device) was resilient to latency whilst other devices see an increase in MT over and above the inherent delay. Figure 4 shows that touchscreen throughput remained high during all latency conditions, whereas other devices dropped at varying rates. Both controller and drawing tablet steadily lost performance as latency increased. Mouse performance started higher than both controller and touchscreen, but also decreased at the fastest rate. The right side of Figure 4 shows IP relative to peak device performance (60ms of latency), highlighting the rate of performance loss with latency. Latency was found to be significantly and negatively correlated with IP (Pearson's R of $-.25$, $p < .001$).

Repeated-measures ANOVA show significant effects of *latency* ($F_{7,105} = 137.02$, $p < .001$, $\eta^2 = .50$) and *device* ($F_{3,45} = 187.56$, $p < .001$, $\eta^2 = .89$) on *IP*, with a significant interaction between *latency* and *device* ($F_{21,315} = 17.06$, $p < .001$, $\eta^2 = .24$).

Follow-up pairwise t-tests with holm adjustment on *IP* and *latency* (using neighbouring points) were performed on each device. For the touchscreen, no pairs were significantly

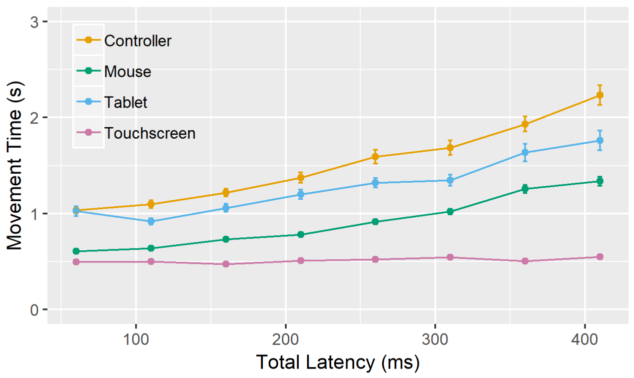

Figure 3: Movement Time (MT) for stationary rounds. (MT = Duration - Latency).

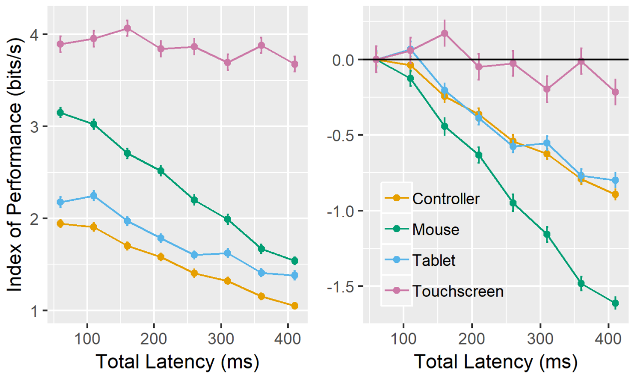

Figure 4: Left: IP of stationary rounds. Right: IP of each device relative to performance at 60ms (black line).

different ($p > .01$). For the mouse, all pairs were different except for the 60:110, 160:210, 260:310 and 360:410 latency pairs ($p > .01$). For both drawing tablet and controller only the 110:160, 210:260 and 310:360 pairs were significantly different (all $p < .01$).

4.2 Path Metrics

Cursor paths are visualized in Figure 5, and are useful for interpreting numeric path metrics in Figure 6. Both movement error (ME) and variability (MV) results look similar, as MV is the standard deviation of ME. Both movement offset (MO), ME and MV produce some overlap at various levels, whereas task axis crossing (TAC), movement direction change (MDC) and orthogonal direction change (ODC) produce clear distinctions between devices. Since the task axis starts at the cursor's start position, small direction changes early in the task can cause a task axis crossing (TAC). Small changes in direction may also cause MDCs, even if the direction is quickly corrected.

By examining both path metrics and path visualizations, we can identify differences between devices. The average controller path strayed far from the task axis (high ME), but had few direction changes (MDC & ODC). The thumbstick

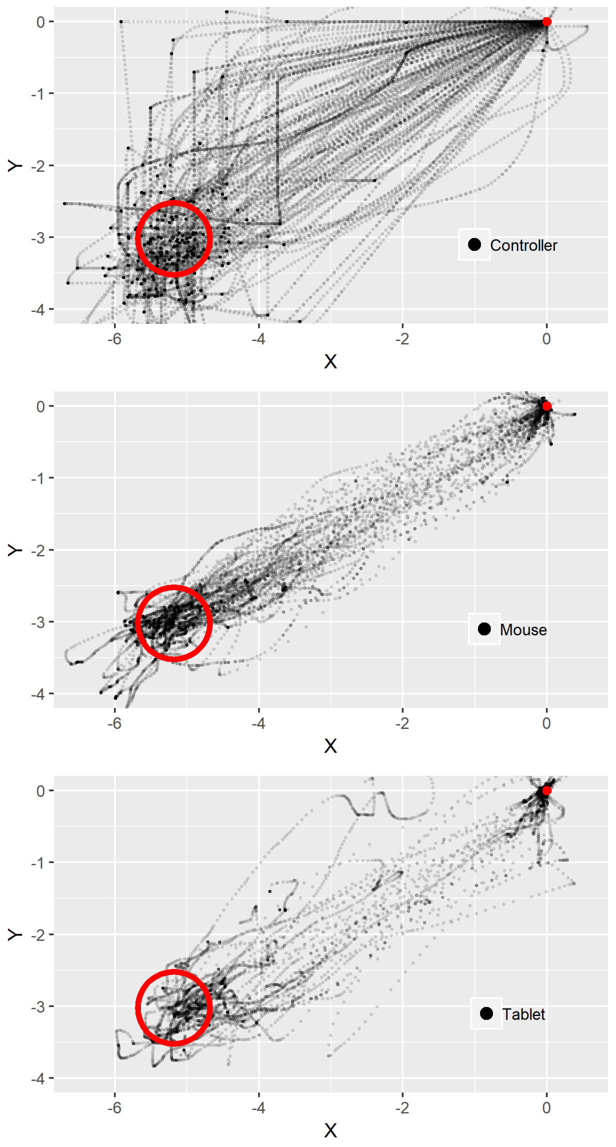

Figure 5: Random selection of 215 cursor paths of stationary rounds with varying latencies (target width 1, target distance 6, ID of 2.81). Cursor starts at red dot in top right, and moves towards red target circle in bottom left.

produced paths with straight sections that often diverged from the optimal path, especially at higher latencies. At low latencies, the controller had few misclicks and target re-entries (TRE), however these rapidly increased when adding more than 50ms of latency.

The average mouse path stayed close to the task axis (lowest ME of all devices), resulting in a high number of task axis crossings (TAC) and direction changes (MDC). Relatively few overshoots (except at latencies > 360ms) led to fewer misclicks and orthogonal direction changes (ODC). The mouse has both the accuracy and precision needed to make minor adjustments to follow optimal paths, showcasing why it is popular for fast-paced shooters. The mouse, however, sees the largest decreased in performance as latency increases.

Drawing-tablet paths were erratic and rarely straight. Overcompensation led to a high number of task axis crossings, and high ME. Short and jerky hand movements produced a high number TRE's and misclicks. Every slight hand movement causes cursor movement, which is exacerbated when hovering. Most users opted to hover the pen over the tablet rather than rest it on the tablet and drag it along, which increases stability. Participants were observed often glancing down at the tablet, moving their hand a bit, then looking back at the screen.

To better understand the effects of latency on our path metrics, we compare relative gains and losses of various path metrics (right subfigures of Figure 6). Interestingly, the mouse had decreased ME and MV as latency increased. Users may have learned exactly how much movement is required to move the cursor exact distances. On the other hand, movement direction changes (MDC) increased steadily with latency due to more corrective actions being needed to keep the cursor near the task axis. The controller had the highest ME and MV gain with latency, as delayed feedback hampered corrective direction changes, resulting in inefficient paths. Latency had little effect on MDC and ODCs. The tablet's ME, MV and MO were relatively unchanged by latency, however, the delayed feedback led to increased direction changes (both MDC and ODC), as well as TREs and misclicks at higher latencies. Paths were inefficient at all latency levels, but became more erratic as latency increased, contributing to target overshoot. Minor adjustments become increasingly difficult with latency, causing many TREs and short jerky movements.

Since each device has its own distinct profile of path metrics, it is useful to analyze which path metrics are most important for pointing performance. We calculated the cross-correlation (Pearson's R) between all path metrics and IP, and found movement error (ME), movement variability (MV) and target re-entry (TRE) (R of $-.47$, $-.49$ and $-.66$ respectively, each $p < .01$) to be the most correlated with IP .

These correlations indicate an ideal pointing device must be able to closely follow the optimal path. Both mouse and drawing tablet are capable of having a low ME since there are no hardware restrictions limiting their movements. The controller stands out, as its force joystick is easiest to move in certain directions (which may explain the prevalence of 45° lines). This problem is exacerbated as thumbsticks wear down. Adjusting settings such as thumbstick 'deadzone' may improve controller performance, but requires additional setup and knowledge.

Most importantly, our ideal pointing device must have low target re-entry (TRE). TRE is most strongly correlated

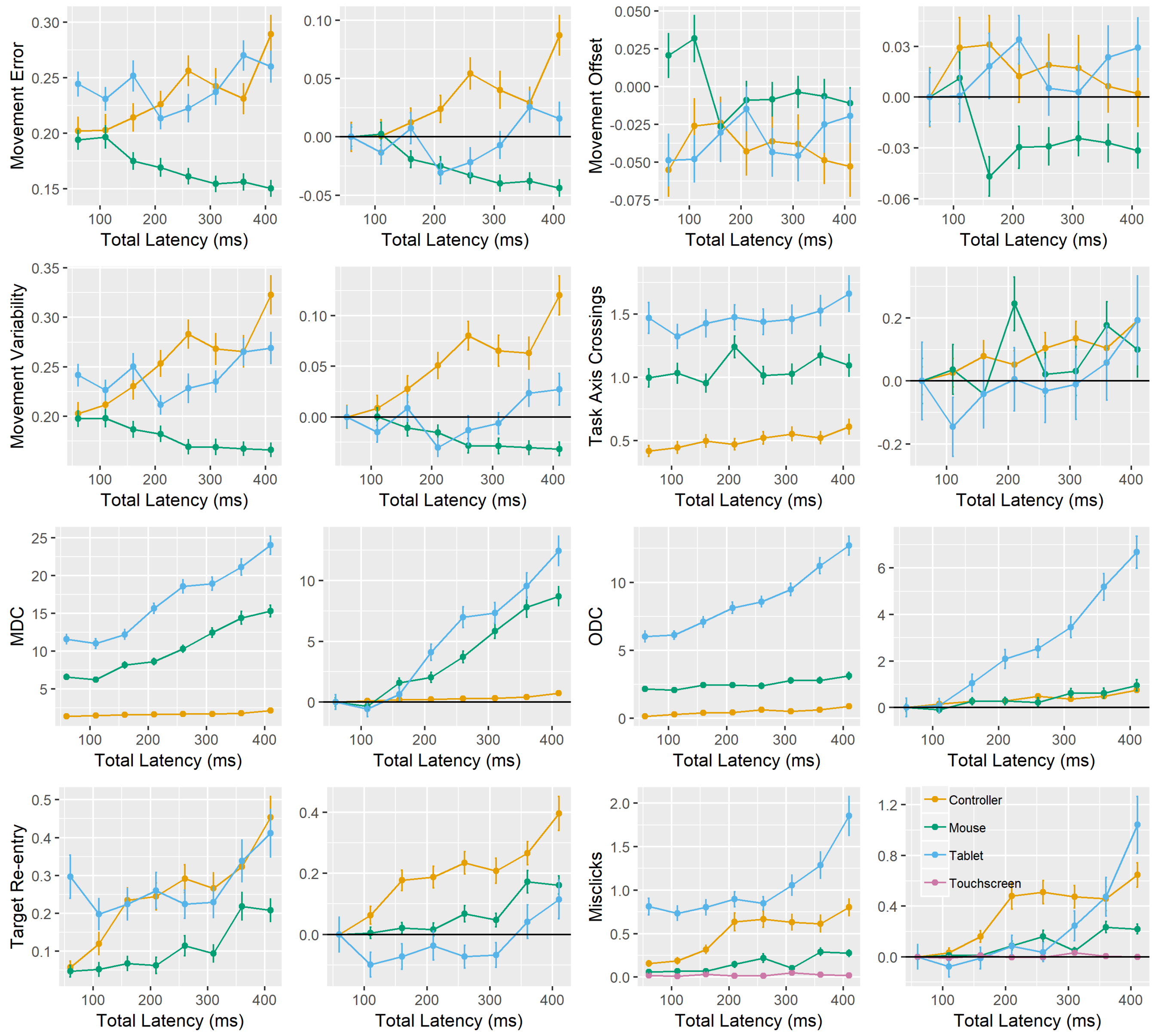

Figure 6: Left of each pair of charts: path metrics for stationary pointing tasks. Right of each pair: path metrics relative to performance at 60ms. From left to right, top to bottom: movement error (ME), movement offset (MO), movement variability (MV), task axis crossings (TAC), movement direction change (MDC), orthogonal direction change (ODC), target re-entry (TRE), misclicks (MC). ME, MO and MV measured in in-game units (1 unit=300 pixels). Touchscreen included for misclicks.

with IP across all devices and latencies (R of −.66), as target overshoot takes time to correct. A high TRE implies the user had difficulty making minor adjustments with the cursor. A good example is the controller: users had less control over cursor speed due to its thumbstick being rate-controlled and displacement-limited. The drawing tablet suffered high TREs due to its high sensitivity.

5 RESULTS: MOVING TARGETS

In moving-target tasks, the target had a speed of 1200 pixels/sec. Movement made the task more difficult, resulting in 3% of the rounds taking the maximum time of 10 seconds (177 of 6144 rounds).

Our results confirm previous findings that the effects of latency are exacerbated with fast game speeds [8, 21]. Latency was found to be significantly and negatively correlated with IP (Pearson's R of −.36, $p < .001$). Repeated-measures

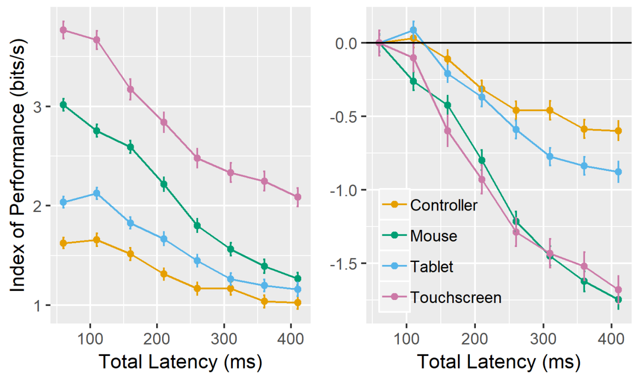

Figure 7: Left: IP of moving rounds. Right: IP relative to performance at 60ms (black line).

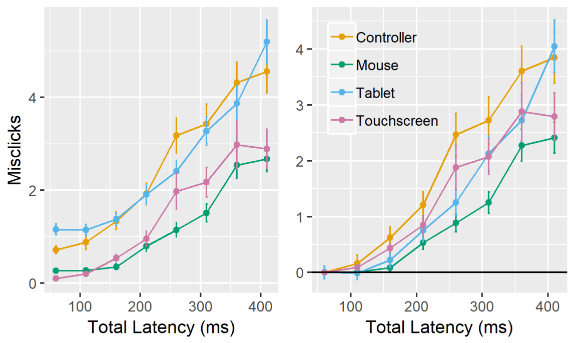

Figure 8: Misclicks of moving rounds. Touchscreen included.

ANOVA shows significant effects of *latency* ($F_{7,105} = 171.42$, $p < .001$, $\eta^2 = .62$) and *device* ($F_{3,45} = 89.60$, $p < .001$, $\eta^2 = .72$) on *IP*, with a significant interaction between *latency* and *device* ($F_{21,315} = 8.32$, $p < .001$, $\eta^2 = .19$).

Follow-up pairwise t-tests with Holm adjustment on *performance* and *latency* (using neighbouring points) were performed on each device. For both touchscreen and controller, no pairs were significantly different ($p < .01$). For the mouse, the following pairs were not significantly different ($p > .01$): 60:110, 110:160, 260:310, 310:360, and 360:410. For the drawing tablet, only the 110:160 pair was significantly different.

The biggest difference in IP between stationary and moving targets was touchscreen performance. Latency had no effect on touchscreen for stationary targets, but had a dramatic effect for moving targets (Figure 7). All three indirect devices (mouse, tablet, controller) had roughly the same IP at 410ms latency.

When examining relative performance losses (right side of Figure 7), IP for both the mouse and touchscreen degraded far more steeply than the controller and touchscreen (though the touchscreen and mouse had a higher starting IP). The controller had relatively low overall performance, but proved to be the most resilient to latency, only losing −.55 bits/s at the highest latency levels.

A moving target meant there was no single task axis, invalidating most path metrics other than target re-entry (TRE)

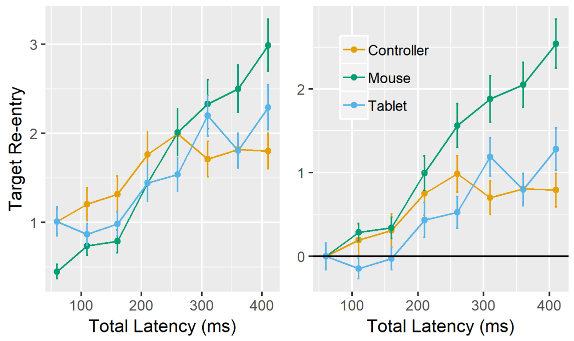

Figure 9: Target re-entries (TRE) of moving rounds.

and misclicks (MC). Both TRE and MC (Figures 8 and 9) rapidly increased with latency, as the target could move from beneath the cursor. Both TRE and MC with moving targets were significantly higher than stationary rounds for all latency levels. The increased misclicks may be due to participants clicking before the cursor arrived at the target, and rapidly clicking at higher latencies (there was no penalty for misclicks in the task).

6 RESULTS: EXPERIENCE QUESTIONNAIRES

Questionnaire results represent both stationary and moving target rounds due to both being present in each latency level. Experience and latency were strongly correlated, with non-touchscreen devices providing similar experiences across all questions (Figure 10). Participants felt less capable (Q1), had less fun (Q2), attributed performance less internally (Q4), became more frustrated (Q5) and felt the cursor was less responsive (Q6) as latency increased. Latency had little effect on effort spent (Q3). Kruskal-Wallis tests showed significant differences between devices on all questions (Q1: $\chi^2 = 1131$, Q2: $\chi^2 = 1187$, Q3: $\chi^2 = 191$, Q4: $\chi^2 = 755$, Q5: $\chi^2 = 838$, Q6: $\chi^2 = 303$, all $p < .001$). All questions (excluding Q5) were negatively correlated with latency (Pearson's R, $Q1 = -.61$, $Q2 = -.50$, $Q3 = -.14$, $Q4 = -.63$, $Q5 = .53$, $Q6 = -.76$, all $p < .001$). The subjective results show that participants were able to clearly distinguish between 100ms of added latency (Q6) by judging cursor responsiveness (they were not told how much latency was added). The touchscreen is rated more positively than other devices, likely due to superior performance with stationary targets (Figures 4 and 7).

After the study was concluded, participants were asked which device they preferred, and which was most resilient to lag, with 13 out of 16 chose the touchscreen for both questions.When asked about strategies, most players mentioned predicting or leading target movement. Several mentioned they clicked more rapidly under high latencies. Some participants adopted a two-handed approach when using the touchscreen, with each hand covering half the screen.

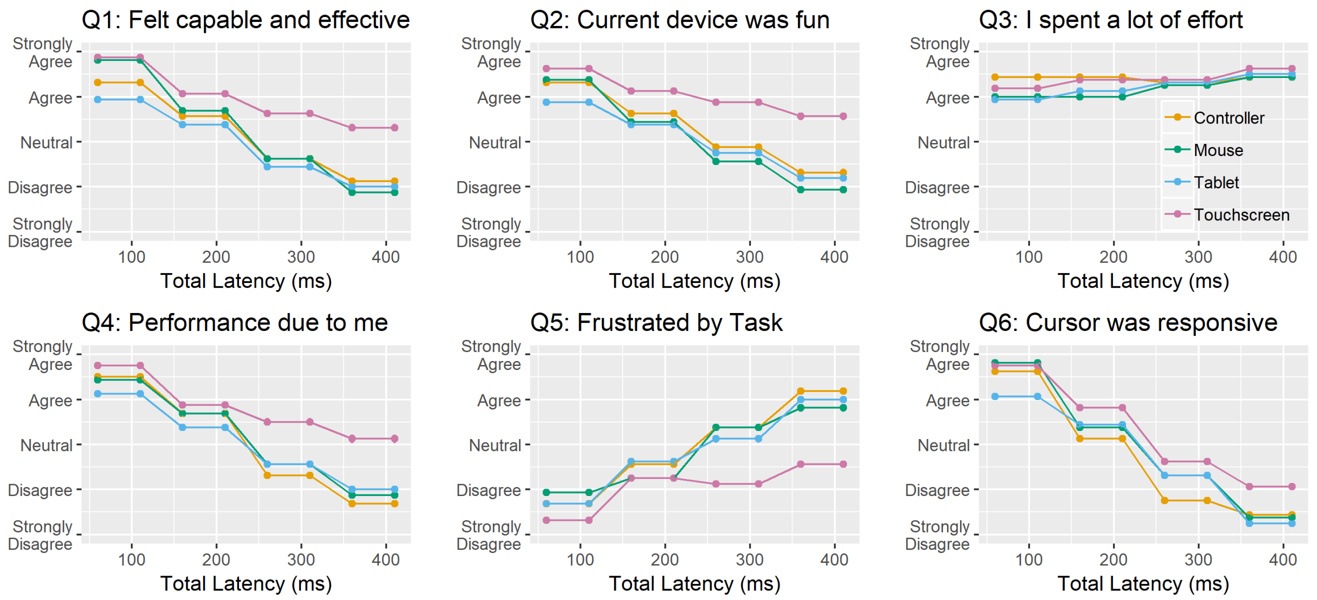

Figure 10: Survey questions on a 5-point Likert scale with standard error bars. Higher means agreement with the question. Survey asked after every second level of latency.

7 DISCUSSION

7.1 Main results for each device

Game Controller. The controller had poor performance (IP), and created straight but inefficient paths that had few direction changes. Overshoot occurred frequently when attempting to select the target, as users had less control over cursor speed. Interestingly, the controller was most resilient to latency regarding throughput (although this may be due to its poor initial performance).

Drawing Tablet. The tablet had poor overall performance, and created erratic and inefficient paths with many slippage mistakes when selecting the target (i.e. target re-entries and misclicks). This may be attributed to participants' inexperience with this device (i.e., tapping without moving the pen horizontally is difficult). It is worth noting most participants treated the drawing tablet as a relative movement device rather than an absolute one, with most participants constantly hovering the pen over the tablet, looking at the screen while making small horizontal movements, only tapping once the cursor was over the target. If the targets were larger, participants with good spatial skills might be able to guess where to tap directly (like a touchscreen), skipping the hovering and correcting phases (as seems to be the case in games like Osu!). This could allow the tablet to be more resilient to lag for stationary targets.

Mouse. The mouse was characterized by good performance and efficient paths with many small direction changes along the task axis. Mouse performance degraded more steeply than other devices as latency increased, but this may be because of this device's higher initial performance. The mouse allows for quick and precise movements using relatively little physical space, and so users may be less likely to overcompensate when lag is present.

Touchscreen. The touchscreen had the best performance, and was unaffected by latency for stationary pointing. The touchscreen also performed well with interception tasks, but performance rapidly decreased with latency. Strong performance overall and freedom from lag effects for stationary targets mean the touchscreen is the best examined device for games that must deal with local latency.

Path Metrics. By analyzing device-specific cursor path metrics, we determined that target re-entry is the single most important path metric to measure. Ideal pointing devices must not be prone to slippage or overshooting. Cursor speed must be easily adjustable, and able to quickly traverse large distances whilst retaining the ability to make small adjustments. The second-most important path metric in relation to throughput is the ability to follow the optimal path (ME). Pointing devices should allow for continuously altering movement direction, and not have mechanical or hardware limitations in directional control.

7.2 Design Recommendations

Aim Assist for Drawing Tablet. The drawing tablet has significant slippage when bringing the pen down for a tap. This can be seen in Figure 5, and leads to many misclicks and target re-entries. Target magnetism or sticky targets could be employed over targets or buttons – these methods slow cursor movement over a target, reducing overshoot [18].

Physical Buttons for Tablets. Physical buttons on the tablet, pen or keyboard can be used instead of virtual buttons to reduce the slippage effects that are exacerbated by local latency. This reduces the number of required pen taps, and allows users to keep the pen on the surface.

Wide Corridors for Controllers. Since the controller mostly produced 45° straight-line paths (that became wider with increased lag), wide corridors with targets on fixed-angle paths from likely starting points could be implemented. Using wide corridors reduces the importance of movement error (ME), and requires fewer direction changes (MDC).

Aim Assist for Controllers. Although already implemented in many console shooter games, we recommend target magnetism or aim assist for fast-paced pointing tasks involving controllers. This helps prevent target overshoot, and has been shown to reduce latency effects in targeting [18].

Use Touchscreens for Pointing Tasks. For both selection and interception tasks, the touchscreen outperformed all other devices. The direct input and absolute positioning of the touchscreen make it ideal for fast-paced pointing in the presence of local latency. Interesting possibilities for making use of direct-touch capabilities include hybrid controllers such as the Wii U's unique touchscreen-and-controller, which allows for relative movement with thumbsticks or absolute positioning using the touchscreen.

Even Low Latency has Significant Effects. Our study found total latencies of 110ms to have a significant effect on stationary selection tasks, and a higher effect on moving target selection tasks. As other studies have shown [15, 16, 18, 31], low latencies are still noticeable by users and *do* significantly effect experience. Care should be taken to reduce processing time and local latency as much as possible.

7.3 Limitations and Future Work

There are several ways in which our study can be expanded to provide better coverage. First, we added latency was at the input device level, and simulating latency at different stages of the input pipeline changes the point at which clicks are detected. Compensation techniques can be used if the amount of lag is known, but the game engine cannot know how much latency is present upstream of it. Second, limits on experiment time prevented us from including dragging as a task. Our interception task was similar to dragging, in that visually tracking the cursor is not very different from tracking a dragged object. Dragging would likely limit touchscreen users to some degree, forcing them to drag their finger across the screen instead of tapping (though in our task users could drag and release to select a target). Third, most of our participants had extensive experience with both mouse and touchscreen, but little with drawing tablets, and only some with gamepads. Self-reported device proficiency scores match overall throughput rankings (Figure 4), which may mean that more experienced users of tablets or controllers could provide different performance results. Fourth, the controller was difficult to design for, as both a maximum movement speed and deadzone had to be decided upon. Tuning these parameters is difficult, and this process is more difficult when lag is present. Finally, most path metrics could not be measured with moving targets. It is difficult to measure the most efficient path to a moving target; a more robust set of path metrics for moving targets would be useful for analyzing moving target selection tasks.

In addition to addressing these experimental limitations, our future work will explore other devices, such as soft joysticks on touchscreens. Being able to simultaneously view both hand and screen may affect performance. In addition, we will look at hybrid pointing devices to see when each input mode is most useful. Hybrid devices include: Wii U gamepad (thumbsticks, touchscreen and motion), and Steam and PS4 controllers (thumbsticks & touchpad). More literal pointing devices include the Wii Mote, PlayStation Move, and Microsoft's Kinect, and provide a direct analogy to pointing tasks by sensing arm movement. These devices are interesting from the perspective of local latency because they use a hybrid between absolute and relative pointing. VR is another interesting setting – one study found hand-tracking in VR to be far more affected by latency than head-tracking [34].

8 CONCLUSION

Local latency affects many computing applications, with games being especially sensitive. It is difficult to study games and latency due to their diversity in genres, tasks and input devices. Common tasks include pointing and interception, and input devices have a variety of properties including direct/indirect input and absolute/relative movement. In this paper we performed a study examining the effects of latency on four different input devices (mouse, touchscreen, gamepad, and drawing tablet) on targeting and interception tasks. Our work shows that latency affects each device differently - the touchscreen was unaffected in stationary targeting whilst other devices suffered. Latency was found to significantly reduce performance and player experience, and also substantially affect cursor paths. By analyzing paths we gained new understanding about the effects of latency, and found target re-entry and movement error to be highly correlated with performance.

REFERENCES

[1] [n. d.]. Human Benchmark - Reaction Time Stats. Retrieved 2018-8-15 from http://www.humanbenchmark.com/tests/reactiontime/stats.php

[2] Glen Anderson, Rina Doherty, and Subhashini Ganapathy. 2011. *User Perception of Touch Screen Latency.* Springer Berlin Heidelberg, Berlin, Heidelberg, 195–202. https://doi.org/10.1007/978-3-642-21675-6_23

[3] 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 3rd ACM SIGCOMM Workshop on Network and System Support for Games (NetGames '04).* ACM, New York, NY, USA, 144–151. https://doi.org/10.1145/1016540.1016556

[4] Mark Claypool. 2016. On Models for Game Input with Delay; Moving Target Selection with a Mouse. In *2016 IEEE International Symposium on Multimedia (ISM).* 575–582. https://doi.org/10.1109/ISM.2016.0125

[5] Mark Claypool. 2018. Game Input with Delay; Moving Target Selection with a Game Controller Thumbstick. *ACM Trans. Multimedia Comput. Commun. Appl.* 14, 3s (jun 2018), 57:1—-57:22. https://doi.org/10.1145/3187288

[6] Mark Claypool and Kajal Claypool. 2006. Latency and player actions in online games. *Commun. ACM* 49, 11 (2006), 40–45.

[7] Mark Claypool and Kajal Claypool. 2010. Latency can kill. *Proceedings of the first annual ACM SIGMM conference on Multimedia systems - MMSys '10* (2010), 215. https://doi.org/10.1145/1730836.1730863

[8] Mark Claypool, Ragnhild Eg, and Kjetil Raaen. 2016. The Effects of Delay on Game Actions. *Proceedings of the 2016 Annual Symposium on Computer-Human Interaction in Play Companion Extended Abstracts - CHI PLAY Companion '16* (2016), 117–123. https://doi.org/10.1145/2968120.2987743

[9] Jonathan Deber, Ricardo Jota, Clifton Forlines, and Daniel Wigdor. 2015. How Much Faster is Fast Enough? User Perception of Latency & Latency Improvements in Direct and Indirect Touch. *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems - CHI '15* 1, 1 (2015), 1827–1836. https://doi.org/10.1145/2702123.2702300

[10] Ansgar E Depping and Regan L Mandryk. 2017. Why is This Happening to Me?: How Player Attribution Can Broaden Our Understanding of Player Experience. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17).* ACM, New York, NY, USA, 1040–1052. https://doi.org/10.1145/3025453.3025648

[11] Matthias Dick, Oliver Wellnitz, and Lars Wolf. 2005. Analysis of Factors Affecting Players' Performance and Perception in Multiplayer Games. In *Proceedings of 4th ACM SIGCOMM Workshop on Network and System Support for Games (NetGames '05).* ACM, New York, NY, USA, 1–7. https://doi.org/10.1145/1103599.1103624

[12] Carl Gutwin. 2001. *The Effects of Network Delays on Group Work in Real-Time Groupware.* Springer Netherlands, Dordrecht, 299–318. https://doi.org/10.1007/0-306-48019-0_16

[13] Carl Gutwin, Steve Benford, Jeff Dyck, Mike Fraser, Ivan Vaghi, and Chris Greenhalgh. 2004. Revealing Delay in Collaborative Environments. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '04).* ACM, New York, NY, USA, 503–510. https://doi.org/10.1145/985692.985756

[14] Sandra G Hart and Lowell E Staveland. 1988. Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research. In *Human Mental Workload*, Peter A Hancock and Najmedin Meshkati (Eds.). Advances in Psychology, Vol. 52. North-Holland, 139–183. https://doi.org/10.1016/S0166-4115(08)62386-9

[15] Niels Henze, Markus Funk, and Alireza Sahami Shirazi. 2016. Software-reduced touchscreen latency. *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services - MobileHCI '16* (2016), 434–441. https://doi.org/10.1145/2935334.2935381

[16] Niels Henze, Sven Mayer, Huy Viet Le, and Valentin Schwind. 2017. Improving Software-reduced Touchscreen Latency. In *Proceedings of the 19th International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI '17).* ACM, New York, NY, USA, 107:1—-107:8. https://doi.org/10.1145/3098279.3122150

[17] Errol R Hoffman. 1992. Fitts' Law with transmission delay. *Ergonomics* 35, 1 (1992), 37–48. https://doi.org/10.1080/00140139208967796

[18] 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. *Proceedings of the ACM CHI'15 Conference on Human Factors in Computing Systems* 1 (2015), 135–144. https://doi.org/10.1145/2702123.2702432

[19] Richard J Jagacinski, Daniel W Repperger, Sharon L Ward, and Martin S Moran. 1980. A Test of Fitts' Law with Moving Targets. *Human Factors* 22, 2 (1980), 225–233. https://doi.org/10.1177/001872088002200211

[20] 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 SIGCHI Conference on Human Factors in Computing Systems (CHI '13).* ACM, New York, NY, USA, 2291–2300. https://doi.org/10.1145/2470654.2481317

[21] Michael Long and Carl Gutwin. 2018. Characterizing and Modeling the Effects of Local Latency on Game Performance and Experience. In *Proceedings of the ACM Symposium on Computer-Human Interaction in Play (CHI Play 2018).* Melbourne, VC, Australia, 13.

[22] Paul M. Fitts. 1992. The Information Capacity of the Human Motor System in Controlling the Amplitude of Movement. *Journal of experimental psychology. General* 121 (1992), 262–269.

[23] I. Scott MacKenzie. 1992. Fitts' law as a research and design tool in human-computer interaction. *Human-computer interaction* 7, 1 (1992), 91–139. https://doi.org/10.1207/s15327051hci0701_3 arXiv:https://doi.org/10.1207/s15327051hci0701_3

[24] I. Scott MacKenzie. 2013. *Human-Computer Interaction: An Empirical Research Perspective* (1st ed.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[25] I. Scott MacKenzie, Tatu Kauppinen, and Miika Silfverberg. 2001. Accuracy Measures for Evaluating Computer Pointing Devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '01).* ACM, New York, NY, USA, 9–16. https://doi.org/10.1145/365024.365028

[26] I. Scott MacKenzie, Abigail Sellen, and William Buxton. 1991. A comparison of input devices in elemental pointing and dragging tasks. *Proceedings of the SIGCHI conference on Human factors in computing systems: Reaching through technology* 1978 (1991), 161–166. https://doi.org/10.1145/108844.108868

[27] I. Scott Mackenzie and Colin Ware. 1993. Lag as a determinant of human performance in interactive systems. , 488–493 pages.

[28] Edward McAuley, Terry Duncan, and Vance V Tammen. 1989. Psychometric Properties of the Intrinsic Motivation Inventory in a Competitive Sport Setting: A Confirmatory Factor Analysis. *Research Quarterly for Exercise and Sport* 60, 1 (1989), 48–58. https://doi.org/10.1080/02701367.1989.10607413

[29] Albert Ng, Julian Lepinski, Daniel Wigdor, Steven Sanders, and Paul Dietz. 2012. Designing for Low-latency Direct-touch Input. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology (UIST '12).* ACM, New York, NY, USA, 453–464. https://doi.org/10.1145/2380116.2380174

[30] Andriy Pavlovych and Wolfgang Stuerzlinger. 2011. Target Following Performance in the Presence of Latency, Jitter, and Signal Dropouts. In *Proceedings of Graphics Interface 2011 (GI '11).* Canadian Human-Computer Communications Society, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 33–40.

http://dl.acm.org/citation.cfm?id=1992917.1992924

[31] Peter Quax, Patrick Monsieurs, Wim Lamotte, Danny De Vleeschauwer, and Natalie Degrande. 2004. Objective and Subjective Evaluation of the Influence of Small Amounts of Delay and Jitter on a Recent First Person Shooter Game. In *Proceedings of 3rd ACM SIGCOMM Workshop on Network and System Support for Games (NetGames '04)*. ACM, New York, NY, USA, 152–156. https://doi.org/10.1145/1016540.1016557

[32] Richard M Ryan, C Scott Rigby, and Andrew Przybylski. 2006. The Motivational Pull of Video Games: A Self-Determination Theory Approach. *Motivation and Emotion* 30, 4 (dec 2006), 344–360. https://doi.org/10.1007/s11031-006-9051-8

[33] Cheryl Savery, T C Nicholas Graham, Carl Gutwin, and Michelle Brown. 2014. The effects of consistency maintenance methods on player experience and performance in networked games. BT - Computer Supported Cooperative Work, CSCW '14, Baltimore, MD, USA, February 15-19, 2014. (2014), 1344–1355. https://doi.org/10.1145/2531602.2531616

[34] Colin Ware and Ravin Balakrishnan. 1994. Reaching for Objects in VR Displays: Lag and Frame Rate. *ACM Trans. Comput.-Hum. Interact.* 1, 4 (Dec. 1994), 331–356. https://doi.org/10.1145/198425.198426

[35] Min Hong Yun, Songtao He, and Lin Zhong. 2017. Reducing Latency by Eliminating Synchrony. *Proceedings of the 26th International Conference on World Wide Web* (2017), 331–340. https://doi.org/10.1145/3038912.3052557