Characterizing and Modeling the Effects of Local Latency on Game Performance and Experience

Michael Long and Carl Gutwin
Department of Computer Science, University of Saskatchewan
Saskatoon, Canada
michael.long@usask.ca, carl.gutwin@usask.ca

ABSTRACT
Studies have shown that local latency – delays between an input action and the resulting change to the display – can negatively affect gameplay. However, these studies report several different thresholds (from 50 to 500ms) where local latency causes problems, and there is still little understanding of the relationship between the temporal requirements of a game and the effects of local latency. To help designers determine how lag will affect their games, we designed two studies that focus on specific atoms of interaction in simple games, and characterize both gameplay performance and experience under increasing local latency. We use the data from the first study to develop a simple predictive model of performance based on the amount of lag and the speed of the game. We used the model to predict performance in the second study, and our predictions were accurate, particularly for faster games and higher levels of lag. Our work provides a new analysis of how local latency affects games, which explains why some game atoms will be sensitive to latency, and which can allow predictive modeling of when playability will suffer due to lag, even without extensive playtesting.

CCS CONCEPTS
• Human-centered computing → Pointing

Author Keywords
Input lag; local latency; game experience; high-speed games.

INTRODUCTION
Many game genres require fast reflexes and split-second decision-making – from fighting games, to first-person shooters, to ‘precision runners’ such as Geometry Dash. One of the main problems for these games is local latency, the lag between the user’s input action and the resulting visual feedback on the screen [15]. Local latency is caused by the combination of input lag, processing delays in the game software, and processing and refresh latency in the display device. Local latency is different from network lag, and affects both single-player and multi-player games. Previous work has shown that local latency can be 250ms or higher in everyday gaming setups [15].

Several studies have examined the effects of different kinds of latency on game performance and game experience. For example, a study of network delay in a racing game saw effects at 150ms and above [23], and studies of FPS games showed that latencies above 100ms [1] and 150ms [8] negatively affected gameplay. Studies of online role-playing games suggest that higher latencies are tolerable (between 150 and 250ms) [8], and Claypool’s survey [5] suggests that third-person games perform well until latency reaches about 500ms. Network latency is important, but so is local latency, with one study concluding that local latency can sometimes exceed that of network latency [28]. Even local latency as low as 41ms can affect targeting in a game setting [15].

These varying thresholds show that different game situations vary substantially in latency tolerance. Some researchers have identified factors that affect this tolerance – for example, Claypool identified a two-dimensional space with dimensions of precision (how accurate a player’s action must be) and time deadline (how quickly an action must occur after a given event) to characterize a game’s latency tolerance [5]. These factors provide a general understanding of how game genres will be affected by delay, but do not give details about the relationship between a specific game’s temporal requirements and the effects of local latency.

This additional information is a critical next step for game designers who build high-speed games (where latency is particularly noticeable and reaction times below 500ms are typically required). Designers need to know how gameplay performance and experience will be affected by local latency, and when their game will become unplayable due to lag. To provide designers with initial information about the relationship between game speed and local latency, we carried out two studies with games that contain simple “atomic” interactions to examine both performance and play experience at different levels of lag. The first game was a ball-return game similar to Pong, and the second was an arcade shooter inspired by Space Invaders. We tested the
games at multiple speeds (characterized by the player’s time to react) to show how the effects of local latency change with temporal requirements. Our studies showed strong effects of local latency on both performance and play experience. In addition, when we normalized game speed using time-to-react, the effects of lag were similar across multiple speeds.

This invariance suggested that if the effects of local latency could be predicted, designers could better understand the sensitivity of different game atoms even without playtesting. Therefore, we used the data from our first study to build a simple predictive model of performance under latency conditions, using a logistic function that matched the sigmoid shape of the performance data. We then tested the model on the data from the second study. For faster game speeds and higher levels of lag, our predictions were within 5% of the empirical values— which could provide valuable information even in the early stages of game design.

Our results provide new insights into how local latency affects gameplay and game experience. We identify player time-to-react as a key link between game speed and latency effects and show that there are consistent effects of latency on both performance and subjective experience. We also provide an initial predictive model of the effects of local latency and achieve good results in a validation test. Even though our model is simple, it isolates several concepts—time to react, floor and ceiling of performance, and the number of tasks carried out simultaneously by the player—that help designers analyze the temporal granularity of their games and understand the specific effects of latency. With further work to validate the models and apply them in real game-design settings, our results can provide a useful new tool that helps designers understand lag sensitivity.

RELATED WORK

Network Latency and Local Latency

Latency—also called lag—is the delay in time between an event (such as a mouse click) and some consequence of that event (such as the movement of the cursor on screen). Latency is known to negatively affect various tasks such as digital sketching [2], collaborative groupware [32], video scrubbing [20], motion tracking [24] and video games [1,5,8,23]. There are several types of latency, with two main types that affect games: network latency and local latency.

Network latency affects interaction in networked multiplayer games and collaborative virtual environments, and has been studied for many years (e.g., [4,7,18,19,24]). A typical network-lag problem is that an opponent’s true location in a game is different from where they appear on the screen, due to transmission delays between game clients [5,8]. Studies of how people respond to network delay have shown several results that provide a starting point for our work. For example: high-accuracy targeting in FPS games is sensitive to latency as low as 60ms [27]; jitter (variance in latency) causes problems primarily for interpretation of smooth motion and streams, whereas latency causes problems for coordination [11]; and humans can accommodate latencies if movement is predictable [25] or delays are visualized [12].

Although network latency is an important problem for games, it is not the type we investigate in this work. The second kind of latency is local latency—the delay between a user’s input action and the resulting visual feedback on the screen [15]. Local latency has been studied less than network latency, but it can also have substantial effects on game performance and experience (and can be as large or larger than network lag [28]). Even small amounts of local latency can be problematic: one study showed that 41ms of latency can cause significant reduction in aiming performance [15].

There are several sources that contribute to local latency [15]. First, delays can arise in retrieving data from the input device (this is often called “input lag”). For example, wired USB mice are often polled at 125Hz or 250Hz, causing 4–8ms of latency (although devices differ widely; gaming mice can poll at up to 1000Hz, and wireless mice can have much higher latencies). Second, display devices can add substantial lag: monitors have refresh rates of 30–144Hz, adding up to 33ms; televisions can add an additional 30–60ms if they perform processing such as motion smoothing. Displays also vary in terms how long it takes a pixel to change colour, with times ranging from 1–12ms [34,35].

Third, the game software can contribute to local latency. Applications operate in discrete frames, with high-speed games running at 60fps or higher, meaning 16ms of delay between updates. The amount of time between updates is the minimum needed to process input, but some games take multiple frames for this task. A survey of real-world gaming setups and games found overall local latencies from 23 to 243ms [15], and in non-gaming environments or cloud-based gaming, latencies can potentially be substantially higher.

Finally, it is important to distinguish our definition of local latency from a networking technique called “local lag,” which slows down local input so that it has the same delay as remote input [30]. As described above, our work is focused on input-to-display lag and does not involve network issues.

Factors Affecting Game Speed

There are several potential factors that contribute to the overall idea of a game’s speed of interaction. The ball-return game used in our first study closely resembles a 2D aiming task. An aiming movement has several phases: a ballistic phase, which is an open-loop action meant to approach an object (in this case, the trajectory of the moving ball); and a controlled phase, a closed-loop action relying on feedback to connect with the target (in our case, the ball) [9]. In classic aiming tasks players know where their target is, but in our case the target is a moving ball, and the player must place their paddle in the path of the incoming ball to prevent it from going past them. Players must predict where the ball will cross the paddle’s horizontal movement axis.
Claypool and colleagues categorize tasks from networked games by measuring them on two separate axes: precision (accuracy required for a task) and deadline (the time within which task must be performed) [5]. Although created for network latency, this framework is a useful starting point when considering local latency. In our study games, the tasks required both high precision, and a tight deadline (between 600ms and 400ms). Games with an ‘avatar’ model, where the player directly controls their character suffer more from latency, as players expect their character to react quickly to their input. Strategy games such as Warcraft 3 [31] are more resilient to latency, because players do not control any one character directly (instead, they direct many units), reducing the expectation for immediate feedback.

One study showed a trade-off between latency and spatial jitter (shaky hands, signal noise) [25]. Different input devices have differing levels of latency, and differing amounts of spatial jitter. Mice have extremely low amounts of latency and jitter, making them well suited for pointing and other tasks requiring precision. The same study found no decrease in performance up to 58ms of latency (using a mouse with a polling rate of 125Hz). A similar study saw performance degradation at 41ms latency [15], but with a 1000Hz mouse.

Models of Human Performance
Predictive models are valuable in design because, once validated, they can provide information about performance without requiring user studies. Researchers have used models to characterize and predict human performance for many years in HCI, since Fitts developed models of human movement in target acquisition [10]. Various equations have since been proposed that predict movement time based on task difficulty (i.e., the size and distance of the target) [19]. Jagacinski et al. provided an extension to model moving targets [16], with Hoffman adding a linear term for latency [14]. Claypool changed the latency term to be exponential [4] (similar to the formalism we propose below).

Other models include the Keystroke-Level Model (KLM), which divides the performance of a task into different operators: physical operators for pressing buttons or pointing to targets, a mental operator for decision-making, and an operator for system response time [3]. Much work has been done on adding to or refining the KLM [18,26], and similar models have been introduced such as GOMS [17]. Other specialized models have been developed for specific tasks. For example, Cockburn and colleagues created a “search, decision, and pointing” model for predicting performance in menu selection tasks. The SDP model can also characterize transitions from novice performance (based primarily on visual search) to expertise (based primarily on spatial memory, governed by the Hick-Hyman Law [6]).

In addition to HCI laws such as Fitts’ or Hick-Hyman, there are specific functions that characterize human performance in certain situations. Of relevance to our work is the “psychometric function,” a sigmoid-shaped curve that describes human performance in signal-detection tasks. Psychometric functions are models describing the relationship between a physical stimulus and forced-choice responses. These models typically involve predicting the critical stimulus thresholds needed to elicit a user response, such as the minimum volume at which a sound can be consistently heard [22]. Psychometric tasks include concepts of a performance floor where nothing can be detected, a transition zone with decreasing error rates, and a performance ceiling. As described below, the data from our studies are well characterized by the psychometric function.

Psychometric functions are usually modelled using the generalized linear model, often with a logistic function. Logistic models are meant to model categorical or binary dependent variables. Wichmann et al. describe using a maximum-likelihood method to estimate parameters of psychometric functions [33]. They also use Monte Carlo simulations to describe the goodness of fit for models.

**STUDY 1: LOCAL LATENCY EFFECTS / INITIAL MODEL**
Our first study gathered data about how local latency affected performance and gameplay in a simple “atom of interaction.” We also use the performance data to build a predictive model of the effects of local latency.

**S1: Game Design**
We built a custom ball-return game based on the arcade classic Pong [36] (Figure 1). Our game has a single main atom of interaction – the player moves the mouse horizontally to control a paddle that intercepts and returns a bouncing ball. This side-to-side movement is common in many games, and if played on PC would use the mouse or keyboard. The player’s paddle width was 6% of the screen width, with the ball taking up 3% of the screen width. The paddle’s movement was tied to the mouse, and could move as quickly as the hand moving the mouse.

![Figure 1. Study game: player controls the white paddle; the player has just returned the ball upwards. White number indicates time remaining; red number shows return streak.](image)

We controlled game speed (ball velocity) and local latency to create our study conditions. The game was a one-player...
version of Pong where the ball bounced back from the top wall (with a random direction change to reduce predictability). We ensured that the ball never bounced off the left or right side of the screen. If the ball went past the player’s paddle, an error was counted.

**S1: Apparatus and Latency Management**
The custom game was built using Unity3D, and the software recorded all study data. The study ran on a Core i7 3.5GHz Windows 10 PC, with Radeon 6970 video, an MSI Interceptor DS100 mouse (1000Hz), and a BenQ 120Hz HD monitor. Mouse sensitivity was set to 800dpi, and Windows mouse acceleration was disabled. The game ran at a constant 120 fps, matching the refresh rate of the monitor.

Following procedures from prior work [15], base local latency was calculated using a 240fps camera (4.17 ms/frame) that recorded both the mouse and the screen. Review of the frames from mouse movement to screen update showed an average of 49ms local latency (using 10 samples). All charts below include this base latency. To add artificial latency, the system stored mouse input for a duration according to the latency level of the study condition.

**S1: Participants**
Eighteen participants (12 male, 6 female, mean age 30) were recruited from a local university. All but one was right-handed, and reported using the mouse in their dominant hand. Eleven played videogames (mean 6.6 hr/week), and seven reported having experienced latency in games.

**S1: Procedure and Study Conditions**
Participants completed informed consent and demographics questionnaires, and performed a reaction-time test [37]. Participants then played 21 rounds of the Pong game: three practice rounds and 18 test rounds. Each test round lasted 70 seconds, with a three-second break between each round to reset the mouse position. Participants were told the goal of the game was to return the ball with their paddle. If the ball got past, it was reset to the middle of the screen, and the new initial trajectory of the ball shown to the player (these initial interactions with the ball were not counted in our measures). Participants could rest as long as they liked when answering survey questions between rounds.

<table>
<thead>
<tr>
<th>Round</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Added Latency (ms)</td>
<td>0</td>
<td>50</td>
<td>250</td>
<td>0</td>
<td>200</td>
<td>100</td>
<td>150</td>
</tr>
<tr>
<td>Total Local Latency (ms)</td>
<td>49</td>
<td>99</td>
<td>299</td>
<td>49</td>
<td>249</td>
<td>149</td>
<td>199</td>
</tr>
</tbody>
</table>

**Table 1. Local latencies for each round; P=practice.**

The study varied two main factors: game speed, and local latency. Each round used one of three speeds, corresponding to the time for the ball to travel the height of the screen: slow (600ms), medium (500ms), and fast (400ms). Faster ball speeds imply more difficult gameplay. Game speeds were presented in order of slowest to fastest (easiest to hardest) because we wanted to show the effects of latency despite the user having practice.

Each round had a level of added local latency (above the 49ms base latency): 0ms, 50ms, 100ms, 150ms, 200ms and 250ms. Latency levels were randomized at the start of the study, and then presented in the same order for each game speed. All participants used the same initially-randomized order of latencies. Table 1 shows the latency for each round (repeated for each game speed, for a total of 21 rounds).

For each combination of game speed and local latency, participants played one practice round and six testing rounds. After each testing round, participants completed a questionnaire that explored the effects of local lag on play experience, using questions from several sources (Table 2).

**S1: Performance Measure**
Our main measure of player performance was the error rate during the testing rounds: the number of misses divided by the total number of attempts (note that there are more attempts in games with higher ball velocities).

**Table 2. Experience questions asked after each testing round.**

<table>
<thead>
<tr>
<th>Question</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1: I felt capable and effective in that round.</td>
<td>PENS: Competence [29]</td>
</tr>
<tr>
<td>Q2: Playing that round was fun.</td>
<td>IMI: Enjoyment [21]</td>
</tr>
<tr>
<td>Q3: I put a lot of effort into that round.</td>
<td>IMI: Effort [21]</td>
</tr>
<tr>
<td>Q4: How well I did during that round was completely due to me.</td>
<td>Attribution [7]</td>
</tr>
<tr>
<td>Q5: During that round, the movement of the paddle was responsive.</td>
<td>Custom</td>
</tr>
<tr>
<td>Q6: I was frustrated during that round.</td>
<td>TLX: Frustration [13]</td>
</tr>
</tbody>
</table>

**Figure 2. Error rate vs. local latency with standard error bars.**

**S1: Performance Results**
Error rates ranged from ~0.06 to ~0.8 (see Figure 2) and were significantly correlated with local latency (Pearson’s R of 0.853, p<.001). The fastest game speed (400ms travel time) had the highest mean error rate (0.56, s.d. 0.29), with the 500ms game next (0.35, s.d. 0.32), and the 600ms game lowest (0.23, s.d. 0.24). Repeated-measures ANOVA showed significant effects of local latency ($F_{1,585} = 542.46$, p<.001, $η^2=0.83$) and game speed ($F_{2,38} = 367.04$, p<.001, $η^2=0.61$) on error rate. ANOVA also showed a significant interaction between local latency and game speed ($F_{10,170} = 39.03$, p<.001,
$\eta^2=0.44$). (We report the effect size for significant RM-ANOVA results as general eta-squared, considering .01 small, .06 medium, and >.14 large [9]).

The slowest (600ms) game speed was resilient to 150ms of latency; the medium (500ms) game speed was resilient to 50ms of latency, and the fastest game speed was immediately affected at 50ms. The error rate showed a ceiling of ~0.8 because some balls will be returned even if the paddle is positioned randomly.

**Combining Speed and Latency → Time to React**

We can combine game speed and local latency into a single factor that represents the amount of time a player has to react to an event. Since local latency reduces this time (feedback about input is delayed), we subtract the local latency amount from the game speed amount; this new factor is called *Time to React* (TTR), measured in milliseconds. Figure 4 shows the error rate using this new factor (the TTR for each game speed). Note that each game speed has a different range on the X axis, with the maximum possible TTR being the game speed with no latency added. In addition, the error rates move upwards to the left (rather than to the right when using simple latency as in Figure 2).

Figure 4 shows characteristic sigmoid curves. To check whether performance in the slowest (600ms) speed would match a sigmoid curve if continued, we carried out a small follow-up trial with added local latency levels (300ms and 350ms); these results confirm that the curves continue in a similar fashion to those shown in Figure 2.

**S1: Experience Questionnaire Results**

Local latency had a substantial impact on almost all player experience questions (see Figure 4), except for Q3 (effort spent). Players felt less capable, had less fun, attributed success more externally and found the paddle less responsive as latency rose. Local latency was significantly correlated with all survey questions ($p<0.05$, Pearson’s R for Q1: -0.67, Q2: -0.68, Q3: 0.19, Q4: -0.51, Q5: -0.73, Q6: 0.48). In addition, many of the questions show the same sigmoid shape described above for performance data.

![Figure 4. Error rates vs. TTR (game speed – latency), by game speed. Note different curve directions compared to Figure 2.](image)

Participants felt less capable and effective as input latency increased (Q1), and fun decreased as latency increased (Q2); these scores mirror the results of internal vs. external attribution of performance (Q4). Latency had little effect on effort, however (Q3). It is possible that because the 400ms game speed was presented last, participants may have been fatigued and more likely to give up at higher amounts of latency. Many participants were sensitive to small amounts of latency, and reliably noticed 50ms (Q5).

![Figure 3. Responses to player experience questions (Table 2), vs. Time to React using 5-point Likert scales.](image)
were not told if latency would be added in each round (or how much) and were instead told most rounds would not have any latency to reduce false-positive reports. It is possible this biased other question responses.

S1: Modeling Local Latency’s Effects

The consistent sigmoid curves seen in the performance data suggest that the relationship between local latency and Time to React could be modeled using the logistic function. Because our results show differences between game speeds, however, we decided to account for these differences in our model. This alteration accounts for the non-linear effect of local latency seen in the data (i.e., increasing local latency has an accelerating effect rather than a linear one). The new measure is Adjusted Time to React (ATTR): Equation 1 shows how ATTR is calculated (all terms in milliseconds). The increasingly negative effect of latency on TTR is encapsulated in an extra term. Higher latencies are weighted more heavily, thus stretching out higher TTR values.

\[ \text{Adjusted TTR} = \text{Gamespeed} - \left( \frac{\text{Latency}}{1000} \right) \]

We applied Equation 1 to our results and found that ATTR grouped the curves better than the standard TTR, especially at higher levels of local latency (see Figure 5).

We can now create a sigmoid curve equation to predict error rates given an ATTR value. This can be done by using a modified logistic function (Equation 2), where e is Euler’s number, k is the steepness of the curve, L is the curve’s maximum value (ceiling), B is the curve’s minimum value (floor) and x0 is the x value of the sigmoid’s midpoint.

\[ f(x) = B + \frac{L - B}{1 + e^{-k(x-x_0)}} \]

To fit a sigmoid curve, we choose values for the four variables: the floor, the ceiling, a midpoint and the steepness. Generally, it can be assumed that users make occasional mistakes (e.g., a floor of -0.05); because more complex games will have higher floors, we set the floor (B) to be 0.05 * number of tasks. The ceiling should be set to the highest rate of errors that can be expected when moving randomly. For our Pong game, the ball and paddle combined take up 9% of the width of the screen; since the chance of hitting the ball doubles if the player is in the correct half of the screen, we set our ceiling (L) to be 0.82. Setting the proper floor and ceiling requires specific implementation knowledge of the game and its complexity.

The midpoint and slope values must characterize the complexity of an interaction atom. We approximate complexity as the number of tasks that the player must perform. The Pong game has a single task: move the mouse to block the incoming ball. For the midpoint (x0), we estimate a base of 130ms plus an additional 130ms per task (130 + 130 * # of tasks). 130ms was chosen as a base because it is roughly half the median reaction speed to a single simple stimulus [37], with each task taking an additional half the usual reaction speed. We theorize that latency will have a more immediate effect the more complex a task is, thus moving the midpoint further to the right in terms of ATTR. The lower the steepness of the curve (k), the straighter the line. Testing values by hand shows a k of 0.05 * 0.5 to be reasonable, where t is the number of tasks in this interaction. Using number of tasks per interaction is only a rough estimate of interaction complexity, but we estimate that in a specific atom of interaction there will only be a few tasks.

![Figure 5. Error rates vs ATTR, with logistic model shown in red dots (B=0.05, L=0.81, x0=260, k=0.025, Equation 2).](image)

Using all game speeds, the model has an R^2 of 0.83 (R^2 of 0.83 for the 600ms game, 0.85 for 500ms, 0.81 for 400ms). We also compared the error rates predicted by the model to the actual rates seen in the study (Table 3). At 300ms ATTR, our predicted error rate was within 1% of the actual error rates (Table 4). Overall mean differences between predicted and actual error rates were low as well, with all game speeds having less than 3% absolute mean difference. This indicates that the model fits our data well – in the next study, we test the model on new data from a different task.

<table>
<thead>
<tr>
<th>Game Speed</th>
<th>Mean Error Rate Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>600ms</td>
<td>0.023</td>
</tr>
<tr>
<td>500ms</td>
<td>0.027</td>
</tr>
<tr>
<td>400ms</td>
<td>0.028</td>
</tr>
</tbody>
</table>

Table 3. Mean absolute differences (predicted – measured).

<table>
<thead>
<tr>
<th>Adjusted Time to React</th>
<th>400ms</th>
<th>300ms</th>
<th>200ms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Measured Error Rate</td>
<td>0.096</td>
<td>0.273</td>
<td>0.679</td>
</tr>
<tr>
<td>Predicted Error Rate</td>
<td>0.077</td>
<td>0.268</td>
<td>0.682</td>
</tr>
<tr>
<td>Difference</td>
<td>0.019</td>
<td>0.005</td>
<td>-0.003</td>
</tr>
</tbody>
</table>

Table 4. Measured vs. predicted error rates by ATTR value.

STUDY 2: MORE COMPLEX GAME, MODEL VALIDATION

To further explore the effects of local latency on performance and experience, and to validate our performance model, we performed a second study on a more complex game.

S2: Game Design

We built a custom space shooter game based on Space Invaders. The game was more complex than the Pong game, requiring the player to divide their attention between evading enemy bullets whilst shooting the green invaders.
The additional cognitive load of multitasking provides a way of testing our model on games that are more complex, as well as exploring the interaction effect of dodging and shooting.

The player controls a ship that moves horizontally by moving the mouse left and right (as quickly as the hand moves). The player shoots the green invaders at the top of the screen by clicking the left mouse button. The main task for the player is avoiding the red bullets moving downwards from the top of the screen. Bullets are spawned at random positions along the entire top of the screen, rather than from the invaders (to maintain consistent difficulty regardless of the number of remaining invaders). The same random seed for generating random numbers was used for each participant. Bullets were created at a constant rate of \(6.66/\text{sec}\) and had uniform velocity (determined by the game speed); bullets took either 900ms, 800ms, 700ms, or 600ms to reach the bottom of the screen. The timer and score for the round were displayed as text on the right side of the screen.

Green invaders were arrayed along the top of the screen in two rows of 11. The player could fire by clicking or holding the left mouse button; player bullets moved upwards at a constant speed (for all conditions), and only one player bullet could be active at a time. When the player’s ship was hit by an enemy bullet, an explosion sound was played, and the ship flashed yellow for 500ms, during which they were unable to shoot. Score increased when a player hit an invader and decreased when their ship was hit. Players were told to prioritize dodging enemy bullets over shooting invaders.

![Figure 6. Space Invaders game. The white ship is controlled by the player. The player must dodge the red bullets whilst shooting the moving green invaders.](image)

**S2: Participants**

Twenty participants (9 female, 11 male, average age 25) were recruited from a local university. All but one was right-handed, with 18 people using the mouse in their dominant hand. Sixteen people played videogames during a typical week (6.1 hrs/week). All but one of the participants reported having experienced lag when playing videogames.

**S2: Apparatus and Latency Management**

The study used a custom Unity3D game, a Core i5 3.2GHz Windows 8.1 PC, an Nvidia GeForce GTX 750 video card, an MSI Interceptor DS100 mouse (1000Hz), and a 27” Asus PG279Q gaming monitor (120Hz, 2560x1440). Mouse sensitivity was set to 800dpi; Windows mouse acceleration was disabled. The Space Invaders game ran at a constant 120 fps, matching the refresh rate of the monitor.

Base local latency was measured in the same way as the first study and was found to be 45ms. All charts below involving latency and TTR have this 45ms of local latency included. Artificial latency was managed in the same way as study 1.

**S2: Procedure and Study Conditions**

Participants completed informed consent and demographics questionnaires, and performed a reaction-time test [37]. Participants then performed the main task of the experiment. Each participant played 24 rounds of the Space Invaders game (four shorter practice rounds, 20 testing rounds of 50 seconds each). An experience questionnaire was given after each testing round (questions seen in Table 5). Rounds were played at one of four game speeds (900ms, 800ms, 700ms, or 600ms, based on the travel time of enemy bullets). Each round also had added latency (Table 6).

<table>
<thead>
<tr>
<th>Question</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1: I felt capable and effective when playing that last round.</td>
<td>PENS: Competence [29]</td>
</tr>
<tr>
<td>Q2: Playing that last round was fun.</td>
<td>IMI: Enjoyment [21]</td>
</tr>
<tr>
<td>Q4: How well I did during that last round was completely due to me.</td>
<td>Attribution Measure [7]</td>
</tr>
</tbody>
</table>

**Table 5. Questions asked after each testing round.**

After each set of six rounds, local latency was reset back to the baseline, and the game speed increased to the next level of difficulty (Table 6). In each group of six rounds, added latency started at zero and increasing in ascending order.

<table>
<thead>
<tr>
<th>Round</th>
<th>1P</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Added Latency (ms)</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td>200</td>
<td>300</td>
<td>400</td>
</tr>
<tr>
<td>Total Local Latency (ms)</td>
<td>45</td>
<td>4</td>
<td>145</td>
<td>245</td>
<td>345</td>
<td>445</td>
</tr>
</tbody>
</table>

**Table 6. Local latency in each round; **P=practice.**

**S2: Measures**

Error rate for the Space Invaders game was calculated as the number of times the player’s ship was hit, divided by the length of the round in seconds, with that sum divided by the expected errors from chance (calculated as the errors that could be expected if the player moved randomly). An error rate less than 1.0 meant the player was performing better than moving randomly. We thought it unlikely for players to do worse than chance, so an error rate of 1.0 acted as a soft ceiling. Low error rates meant the player was performing well and was not being hit by many bullets, and a high error rate meant the player was performing poorly. The expected errors from chance was calculated as the combined width of both the player ship and enemy bullets,
multiplied by the spawn rate of enemy bullets per second (6.66/second).

**S2: Performance Results**

As shown in Figure 7, error rates follow sigmoid curves (although less so than in S1), with performance ranging from ~0.10 to ~1.0 errors/sec. The Space Invaders game was, however, more sensitive to local latency: 100ms of added local latency had an immediate effect on errors for all speeds.

![Figure 7. Error rate vs TTR, by game speed. The black line is the expected error rate for random movement.](image)

Error rate was significantly correlated with latency (Pearson’s R of 0.76, p<.001). Repeated-measures ANOVA again showed significant effect of local latency ($F_{4,76}=189.19, p<.001, \eta^2=0.66$) and game speed ($F_{3,57}=36.23, p<.001, \eta^2=0.18$) on error rate, with a significant interaction between latency and game speed ($F_{12,228}=3.34, p<.001, \eta^2=0.07$). The fastest (600ms) game had the highest mean error rate (0.63, s.d. 0.38), followed by 700ms (mean 0.57, s.d. 0.35), 800ms (mean 0.46, s.d. 0.34), and 900ms (mean 0.38, s.d. 0.32).

Follow-up pair-wise t-tests on 900ms game speed with error rate and latency (using neighboring points only) all show significant differences (p<0.01) except for the following TTR pairings: 455:555 (p=0.02), 655:755 (p=0.09), and 755:855 (p=0.16). For the 800ms game, all neighbors were significantly different except 655:755 (p=0.14). For the 700ms game, all were different except 255:355 (p=0.88) and 555:655 (p=0.88). For the 600ms speed, only the 355:455 pairing was significantly different (p<0.01).

There was greater variance in performance for the Space Invaders game than for the Pong game. This is likely due to the random placement of enemy bullets, and increased complexity (dodging & shooting). Over a longer play time, it is possible that, the randomness of bullet placement would eventually average out, providing a more uniform error rate. More complex games may feature more elements beyond the player’s control, likely increasing performance variance.

![Figure 9. Error Rates with Adjusted TTR (Equation 1).](image)

We also applied our Adjusted TTR measure to the performance data; as shown in Figure 9, ATTR again brings the curves for the different game speeds much closer together (although not as close as in study 1). Error rates grouped together more closely at lower ATTR and spread out at high ATTR. It is possible that participants were not given enough practice time after each game speed increase and did not become expert users in that time.

**S2: Player Accuracy Results**

Accuracy was calculated as the number of invaders hit divided by the total number of shots for the round. Local latency was not significantly correlated with accuracy.

![Figure 8. Responses to player experience questions (table 7), vs. Time to React. Questions were scored on a 5-point Likert scale.](image)
(Pearson’s R of -0.02, p=0.64), nor was game speed (R of 0.06, p=0.24). This is likely due to the design of the game: with 22 invaders moving at the top of the screen, holding down the fire button will initially result in many hits; but when few invaders remain, most shots will miss, and a degree of actual aiming is required to score hits. Players may also have differentially prioritized shooting and dodging.

**S2: Experience Questionnaire Results**

As in study 1, all survey questions were negatively correlated with latency (Figure 9); players felt less effective (Q1), had less fun (Q2), and were more likely to attribute their performance externally (Q3) as latency increased. Latency was significantly correlated with all questions (p<0.05), with a Pearson’s R of -0.60 for Q1, -0.57 for Q2 and -0.63 for Q3.

**S2: Evaluation of the Predictive Model**

We used the logistic function model developed from study 1 to predict error rates in study 2. We first analysed the Space Invaders game to determine appropriate coefficients for the model. Space Invaders is composed of two tasks: dodging enemy bullets and shooting invaders. We set the center x position of the sigmoid \(x_0\) to be 130+(2*130)=390ms. The steepness of the slope \(k\) was 0.05*0.5^2=0.0125. The floor was chosen to be 0.1 since there were two tasks (0.05*2), and since we already incorporated the errors due to chance in the error rate itself we set the ceiling at 0.95. The resulting model is shown in red dots in Figure 10.

We calculated the goodness of fit of our model for study 2 data, and also considered the actual error of the predictions. Using data from all game speeds, the correlation of the model to the data gives an \(R^2\) of 0.65, with an \(R^2\) of 0.50 for the 900ms game, 0.72 for 800ms, 0.70 for 700ms, and 0.71 for 600ms. These are lower than the \(R^2\) values of the model from the first study. \(R^2\) represents how much variance our model explains, but since the bullet placement was random, there is some variance our model does not explain. Task complexity may also contribute to the lower fit.

![Figure 10. Error rates by game speed, with predicted error rate shown in red dots (calculated with Equation 2).](image)

Predicted error rates matched the measured error rates reasonably well when ATTR was low (left-hand side of Figure 10), but less well when ATTR was high. At 500ms ATTR, predicted and actual Error Rate means differed by close to 0.1, with overall mean error rate differences for different game speeds between 0.144 (800ms game speed) and 0.183 (700ms game speed).

<table>
<thead>
<tr>
<th>Game Speed</th>
<th>500ms</th>
<th>375ms</th>
<th>250ms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Measured Error Rate</td>
<td>0.367</td>
<td>0.591</td>
<td>0.802</td>
</tr>
<tr>
<td>Predicted Error Rate</td>
<td>0.270</td>
<td>0.562</td>
<td>0.849</td>
</tr>
<tr>
<td>Difference</td>
<td>0.097</td>
<td>0.029</td>
<td>-0.047</td>
</tr>
</tbody>
</table>

Table 7. Difference between measured and predicted error rates, averaged across all Game Speeds for certain ATTR (i.e., the distance between the red and other curves in Figure 10).

Looking at Figure 10, we can see that the estimated steepness of the curve \(k\) was too high and needs to be flattened out more, and the midpoint \(x_0\) should be shifted over to the right a bit. It appears that the complexity of a game comprises more than just equally-weighted tasks, and these tasks may have their own interaction effects. More testing needs to be done on other games to improve the estimation of slope \(k\), especially in games with multiple tasks.

**DISCUSSION**

Our two studies provide several main results:

- Player performance was strongly affected by latency, although sensitivity to latency depends on game speed, with faster-paced games affected by lower latencies;
- Player experience was also strongly affected, with substantial changes to enjoyment, frustration, perceived competence, and attribution as latency increased;
- The measure of Time to React (game speed – local latency) can help to normalize the differences between different game speeds;
- Both performance and experience data reflect sigmoid curves as latency increases, with readily-apparent floor, ceiling, and transition zones;
- Error rates can be predicted with moderate accuracy using a modified logistic function: with a model built from study 1 data, our predictions about error rates in study 2 were within 5-10% for faster game speeds (although less accurate for slower game speeds).

In the following sections, we present observations of and comments from the studies, consider how our results and models can be applied in other game-design settings, and review potential limitations of our work.

**Observations from the Studies**

After both experiments, we asked participants if they had any comments about the study or about the experience of playing the game with local latency. In study 1, several
participants mentioned that as lag increased, it became an entirely different game – with high local latency, it became important to intercept the ball as it was about to cross the goal line, instead of leisurely placing the paddle in the path of the oncoming ball. Participants also mentioned that they needed to use more of an open-loop motion at high lag, rather than an interactive closed-loop motion (which is dependent on visual feedback). Several people also mentioned that they enjoyed the higher lag levels (though as can be seen in Figure 4, these participants were in the minority). The enjoyment of lag may have arisen because of the simplicity of the game itself – that is, lag added a level of challenge to an easy game.

For both studies, when local latency rose above 300ms, some participants expressed that the game was “purely random.” Several participants mentioned that if they encountered latency at such high levels when gaming, they would have quit playing or changed their setup to reduce the lag.

**Application of Predictive Models**

Although we plan to carry out further testing of our logistic model before asking designers to use it, it is important to show that the model can be integrated into a design process. Therefore, here we set out the steps that a designer would carry out to apply our model in a real game-design situation. The first step is to decide what atom of gameplay or interaction is to be analyzed. Games are typically made up of many such atomic interactions, and designers often have intuitions about which atoms must support high-speed play. Most atoms can be categorized as pointing, tracking, reaction or selection. The chosen atom should be simple and repeatable: examples include dodging a grenade thrown in a shooter game, jumping over pits in infinite runners, or blocking or dodging an attack in a fighting game.

Second, the game speed must be measured; this is how quickly the user must act to prevent an error from occurring. The third step is to consider what level of local latency can be expected in typical deployment (e.g., using the analysis presented by Ivkovic [15]. Note that latency does not need to be sensed during eventual gameplay: these estimates are only to guide the designer in determining which game atoms will be impaired by expected latency levels. Overall, the simplicity of testing multiple lag values through a model means designers can explore a wider range of potential scenarios than they could with playtests.

Fourth, the designer calculates Adjusted TTR using game speed and expected local latency using Equation 1. Fifth, the designer needs to determine the four parameters for the sigmoid curve (Equation 2), with the floor chosen based on domain knowledge about the game. Sixth, predictions can be made about likely game performance by plugging in ATTR values to the parameterized version of Equation 2.

In summary:

1. Choose the interaction atom to analyze
2. Determine game speed for this interaction
3. Determine local latency values for investigation
4. Calculate Adjusted TTR using Equation 1
5. Choose parameters for sigmoid curve (Equation 2)
   - Ceiling (B): 0.95, or expected errors from chance
   - Floor (L): # of tasks * 0.05
   - Sigmoid midpoint (x₀): 130 + 130 * # of tasks
   - Steepness of curve (k): 0.05 * 0.5^# of tasks
6. Calculate predicted error rate by plugging in ATTR as x into the parameterized version of Equation 2

Predicted error rates are only part of the effects of local latency, and more work needs to be done to explore the model’s ability to predict other forms of gameplay. We hope that learning how resilient interactions are to local latency will reduce the amount of playtesting needed for fine-tuning a game (even in early stages of design). There are obviously many more factors to take into consideration when designing a game; nevertheless, every game is composed of many simple atoms of interactions that can be individually analyzed. To the best of our knowledge, there have not been any predictive models for latency and error rate in videogames. This work is an initial exploration, but with more analyses and verification tests, a robust and broadly applicable model can be produced. Ideally, the output of our model would be game settings such as game speed, but this would require making game-specific models. It is also possible that this predictive latency-error rate model could be applied to other tasks outside the realm of videogames.

**Limitations**

For the first study, our main experiment did not include 300ms and 350ms latency conditions. While we did a small follow-up study that examined these, performing a larger study would help to validate the data. For the Pong game, players had to predict the point at which the ball would cross the horizontal movement axis of their paddle. It would be interesting to see if performance is affected if the current trajectory of the ball is always shown. This would remove all elements of guessing on the player’s part.

For the Space Invaders study, we would have added more latency levels (between the tested levels) if time had permitted (such as 50ms intervals instead of 100ms intervals), but despite allowing for breaks, participants were getting tired of gameplay after playing for 20 minutes. Adding a 50ms latency level would likely have shown the floor value of our error-rate measure more clearly. We also hypothesize that error rates would stabilize (at the rate of errors due to chance) if higher latency levels were added.

When choosing an atom of interaction to model, it is important to note what can be measured. Not all atoms use an error rate, but most have a measurable outcome (e.g. time, score). The model could be adapted to these other outcomes.
Another factor to consider is that neither of the main studies emulated any jitter (variation in latency). This corresponds to what is likely to occur in real-world settings: while network latency often has considerable amounts of jitter, the sources of local lag (input devices, software processing, and display devices) are typically much less variable. Follow-up work should be done to see if our *Time to React* measures extend to network latency with significant amounts of jitter.

**Future Work**

There is much work yet to do to apply these findings to real game design. Interaction effects must be explored to see whether the effects of local lag, and the efficacy of our models, extend to other atoms. Additionally, we plan to test our results with different input device (e.g., gamepads, keyboards, and touchscreens) with different properties such as indirect/direct input and absolute/relative movement. Third, we will extend our models to determine whether they can predict experience as well as performance, since the response curves for experience measures had similar shapes to the performance curves. Finally, we will carry out testing of our models with designers, to determine the value of our approach in real-world settings.

**CONCLUSION**

Studies have reported several different thresholds for the effects of local latency on game performance and experience (from 50ms to 500ms), suggesting that different games are highly variable in their sensitivity to lag. However, there is still little understanding of the specific relationship between the temporal requirements of a game and the amount of lag. To help game designers understand this relationship, we carried out two studies that varied interaction speed and amount of local latency, and recorded performance and player experience measures. Our results provide new insights into how latency affects games, and suggest a normalized measure of “time to react” that combines game speed and amount of local latency. We developed a predictive model based on a logistic function that can predict game performance with reasonable accuracy, particularly for faster games. Our work shows how game designers can analyze the temporal granularity of their games and use this information to better understand the likely effects of delay.

**ACKNOWLEDGEMENTS**

We thank the anonymous reviewers for their feedback.

**REFERENCES**


