

EFFECTS OF LOCAL LATENCY ON GAMES

A Thesis Submitted to the
College of Graduate and Postdoctoral Studies
in Partial Fulfillment of the Requirements
for the degree of Master of Science
in the Department of Computer Science
University of Saskatchewan
Saskatoon

By
Michael Long

©Michael Long, April 2019. All rights reserved.

PERMISSION TO USE

In presenting this thesis/dissertation in partial fulfillment of the requirements for a Postgraduate degree from the University of Saskatchewan, I agree that the Libraries of this University may make it freely available for inspection. I further agree that permission for copying of this thesis/dissertation in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my thesis/dissertation work or, in their absence, by the Head of the Department or the Dean of the College in which my thesis work was done. It is understood that any copying or publication or use of this thesis/dissertation or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Saskatchewan in any scholarly use which may be made of any material in my thesis/dissertation.

DISCLAIMER

Reference in this thesis/dissertation to any specific commercial products, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement, recommendation, or favoring by the University of Saskatchewan. The views and opinions of the author expressed herein do not state or reflect those of the University of Saskatchewan, and shall not be used for advertising or product endorsement purposes.

Requests for permission to copy or to make other uses of materials in this thesis/dissertation in whole or part should be addressed to:

Head of the Department of Computer Science
University of Saskatchewan
176 Thorvaldson Building, 110 Science Place
Saskatoon, Saskatchewan S7N 5C9 Canada

OR

Dean
College of Graduate and Postdoctoral Studies
University of Saskatchewan
116 Thorvaldson Building, 110 Science Place
Saskatoon, Saskatchewan S7N 5C9 Canada

ABSTRACT

Video games are a major type of entertainment for millions of people, and feature a wide variety of game genres. Many genres of video games require quick reactions, and in these games it is critical for player performance and player experience that the game is responsive. One of the major contributing factors that can make games less responsive is *local latency* — the total delay between input and a resulting change to the screen. Local latency is produced by a combination of delays from input devices, software processing, and displays. Due to latency, game companies spend considerable time and money play-testing their games to ensure the game is both responsive and that the in-game difficulty is reasonable. Past studies have made it clear that local latency negatively affects both player performance and experience, but there is still little knowledge about local latency’s exact effects on games. In this thesis, we address this problem by providing game designers with more knowledge about local latency’s effects. First, we performed a study to examine latency’s effects on performance and experience for popular pointing input devices used with games. Our results show significant differences between devices based on the task and the amount of latency. We then provide design guidelines based on our findings. Second, we performed a study to understand latency’s effects on ‘atoms’ of interaction in games. The study varied both latency and game speed, and found game speed to affect a task’s sensitivity to latency. Third, we used our findings to build a model to help designers quickly identify latency-sensitive game atoms, thus saving time during play-testing. We built and validated a model that predicts errors rates in a game atom based on latency and game speed. Our work helps game designers by providing new insight into latency’s varied effects and by modelling and predicting those effects.

ACKNOWLEDGEMENTS

I would first like to thank my supervisor Carl Gutwin, for helping and guiding my research, as well as his his no-nonsense approach of cutting through flowery prose and getting right to the point. I also wish to thank my colleagues both in and outside of the HCI Lab for being guinea pigs in my studies, and for providing a fun and friendly work environment. I also extend my thanks to my family — thanks goes to my dad Richard and mom Patricia for showing me that higher-education can lead to a great life. I also want to thank my two brothers Jeffrey and Jeremy for first showing me that computers are pretty awesome.

CONTENTS

Permission to Use	i
Abstract	ii
Acknowledgements	iii
Contents	iv
List of Tables	vii
List of Figures	viii
List of Abbreviations	x
1 Introduction	1
1.1 Motivation	1
1.2 Problem	2
1.3 Solution	3
1.4 Steps to a Solution	4
1.4.1 Effects of Latency on Input Devices	5
1.4.2 Effects of Latency and Game Speed on Game Atoms	6
1.4.3 Modelling Error Rates	7
1.5 Evaluation	8
1.5.1 Effects of Latency on Input Devices	8
1.5.2 Effects of Latency and Interaction Speed on Game Atoms	9
1.5.3 Modelling Error Rates	10
1.6 Contribution	10
1.7 Thesis Outline	12
2 Related Work	14
2.1 Latency and Games	14
2.1.1 Local Latency	15
2.1.2 Network Latency	17
2.1.3 Methods for Combating Latency	20
2.2 Input Devices	22
2.2.1 Pointing Device Characteristics	22
2.2.2 Input Devices and Latency	25
2.3 Games and Performance	29
2.3.1 Models of Human Performance	29
2.3.2 Game Atoms of Interaction	32
2.3.3 Subjective Player Experience	33
3 Effects of Latency on Pointing Devices (Study 1)	35
3.1 Methods	37

3.1.1	Input Devices Chosen For The Study	37
3.1.2	Cursor Path Metrics	37
3.1.3	Participants	39
3.1.4	Apparatus	39
3.1.5	Procedure	40
3.1.6	Design	40
3.2	Results: Stationary Targets	42
3.2.1	Movement Time	42
3.2.2	Path Metrics	43
3.3	Results: Moving Targets	47
3.4	Results: Experience Questionnaires	49
3.5	Summary	50
4	Effects of Local Latency on Game Speed and Game Atoms (Study 2)	53
4.1	Identifying Game Atoms	54
4.2	Methods	55
4.2.1	Game Design	55
4.2.2	Apparatus and Latency Management	56
4.2.3	Participants	56
4.2.4	Procedure and Study Conditions	57
4.2.5	Performance Measure	58
4.2.6	Performance Results	58
4.2.7	Combining Speed and Latency into Time to React	58
4.2.8	Experience Questionnaire Results	60
5	Building an Error-Rate Prediction Model	63
5.1	Creating a Predictive Model	64
5.2	Application of Predictive Model	67
5.2.1	Steps to Apply Model	67
6	Model Validation (Study 3)	69
6.1	Methods	70
6.1.1	Apparatus and Latency Management	70
6.1.2	Participants	71
6.1.3	Procedure and Study Conditions	71
6.1.4	Performance Measure	72
6.2	Results	72
6.2.1	Player Accuracy Results	74
6.2.2	Experience Questionnaire Results	74
6.2.3	Evaluation of Predictive Model	76
7	Discussion	79
7.1	Summary of Main Findings	79
7.2	Pointing Devices and Latency	80
7.2.1	Performance Explained by Cursor Paths	80
7.2.2	Effects of Latency on Cursor Paths	82
7.2.3	Lessons for Designers	83
7.2.4	Relevance of Findings in the Real World	85

7.3	Game Atoms, Game Speed and Latency	86
7.3.1	Game Speed Affects Performance and Experience	86
7.3.2	Time to React	87
7.3.3	Latency's Effects Are Exponential	88
7.4	Predictive Error Rate Model	89
7.4.1	Error Rates Followed Logistic Functions	89
7.4.2	Usefulness of the Model	90
7.5	Applications Outside Gaming	92
7.6	Limitations	93
7.6.1	Study Design	93
7.6.2	Latency Simulation	94
7.7	Future Work	95
7.7.1	Hybrid Input Devices	95
7.7.2	Predicting Player Experience	96
7.7.3	Jitter	96
7.7.4	Regression and Data-Driven Parameters	96
7.7.5	Error Rate Model Validation	97
8	Conclusion	98
	References	100
	Appendix A Forms	107
A.1	Consent Form (Study 1)	107
A.2	Consent Form (Study 2)	108
A.3	Consent Form (Study 3)	109
A.4	Study Script and Demographics Questions, Study 1	110
A.5	Study Script and Demographics Questions, Study 2	111
A.6	Study Script and Demographics Questions, Study 3	112
A.7	Example In-Game Questionnaire Screen, Study 2	113

LIST OF TABLES

3.1	In-game experience questions asked after every second latency level.	41
3.2	Local latencies per round; P=practice. S=survey after round. Each level repeated 24 times per device.	41
3.3	Pearson's R cross-correlations between path metrics and IP. All $p < .05$ except for entries denoted by *. Top row, left to right: Index of Performance (IP), Movement Error (ME), Movement Offset (MO), Movement Variability (MV), Task Axis Crossings (TAC), Target Re-entry (TRE), Movement Direction Change (MDC), Orthogonal Direction Change (ODC), Misclicks.	47
3.4	Participants were asked which device they preferred using, and which was most resilient to lag.	51
4.1	Local latencies for each round; P=practice.	57
4.2	Experience questions asked after each testing round.	58
5.1	Mean absolute differences (predicted - measured).	66
5.2	Measured vs. predicted error rates by ATTR value.	66
6.1	Questions asked after each testing round.	72
6.2	Local latency in each round; P=practice.	72
6.3	Mean absolute differences between measured error rates and predicted error rates at different Game Speeds.	76
6.4	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 6.6).	77

LIST OF FIGURES

2.1	Each step in the process of gathering input to displaying output adds delay known as local latency.	15
2.2	A typical cloud gaming setup. At the left side the user sends input over the internet to a server in the cloud. On the server the game scene is then rendered, and to save bandwidth, encoded. The encoded video is then sent back over the internet to be decoded by the user's client [15].	19
2.3	Input devices, sorted by properties. Top left: smartphone with touchscreen, stylus and touchscreen with Nintendo 3DS, touchscreen of Microsoft Surface Studio. Top right: drawing tablet (absolute mode), WiiMote. Bottom right: laptop pointing stick, laptop track pad, thumbsticks on gamepad, mouse, joystick. Bottom left: soft joysticks on touchscreen.	24
2.4	MSI Interceptor DS B1 gaming mouse.	26
2.5	Xbox 360 wired USB gamepad.	26
2.6	Dell 27" LED touchscreen monitor.	27
2.7	Wacom Intuos CLS 480s drawing tablet.	28
2.8	Example s-shaped sigmoid curve with X centered at the inflection point.	31
3.1	Input devices used, divided by device properties.	38
3.2	Cursor (pink) and its path (yellow) along task axis (red) towards target (white circle). This path features one task-axis crossing, and two movement direction changes (green).	38
3.3	Response time histogram of participants in milliseconds (ms).	39
3.4	Experimental setup using touchscreen.	40
3.5	Movement Time (MT) for stationary rounds. (MT = Duration - Latency).	42
3.6	Left: IP of stationary rounds. Right: IP of each device relative to performance at 60ms total latency (black line).	43
3.7	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.	44
3.8	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.	45
3.9	Left: IP of moving rounds. Right: IP relative to performance at 60ms (black line).	48
3.10	Misclicks of moving rounds. Touchscreen included.	49
3.11	Target re-entries (TRE) of moving rounds.	49
3.12	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.	50
4.1	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.	55
4.2	Response time histogram of participants in milliseconds (ms).	56
4.3	Error rate vs. local latency with standard error bars.	59

4.4	Error rates vs. TTR (game speed minus latency), by game speed. Note different curve directions compared to Figure 4.3.	59
4.5	Error rates of small follow-up trial with added local latency levels 300ms and 350ms. Note the lines extend further to the left when compared to Figure 4.4.	60
4.6	Responses to player experience questions (Table 4.2), vs. Time to React using 5-point Likert scales.	61
5.1	Error rates vs. Time to React of study 2. Note the S-shaped sigmoid curves. Reproduced from Figure 4.4 in chapter 4.	64
5.2	Error rates vs ATTR, with logistic model shown in red dots ($B = 0.05$, $L = 0.81$, $x_0 = 260$, $k = 0.025$, Equation 5.2).	65
6.1	Space Invaders game. The white ship is controlled by the player. The player must dodge the red bullets whilst shooting the moving green invaders. The red number on the right is the player's score (enemies hit minus number of times player was hit), and the white number is time remaining in round.	70
6.2	Response time histogram of participants in milliseconds (ms).	71
6.3	Error rate vs TTR, by game speed. The black line is the expected error rate for random movement.	73
6.4	Error Rates with Adjusted TTR (Equation 5.1).	74
6.5	Responses to player experience questions (Table 6.1), vs. Time to React. Questions were scored on a 5-point Likert scale.	75
6.6	Error rates by game speed, with predicted error rate shown in red dots (calculated with Equation 5.2).	77
7.1	Random selection of cursor paths from controller, mouse and drawing tablet trials (reproduction of Figure 3.7 from study 1, chapter 3).	81
7.2	Example of large targets on fixed-angle paths (arranged in grid pattern) seen in Windows 8 Metro UI.	84
7.3	Left: Error rates of multiple game speeds with latency on X axis. Right: Error rates of multiple game speeds with Time to React on X axis. Both figures using results from study 2 (reproduction of Figures 4.3 and 4.4 from chapter 4).	88
7.4	Left: Error rates with a linear term for Time to React. Right: Error rates with quadratic term for TTR. Both figures using results from study 2 (reproduction of Figures 4.4 and 5.2 from chapter 4).	89
7.5	Example sigmoid curve featuring a floor (left), transition area (denoted in red), and ceiling (right). Reproduced from Figure 2.8 in chapter 2.	90
7.6	Error rates from study 2 (chapter 4), with model-predicted error rates in red. Reproduced from Figure 5.2 in chapter 4.	91
7.7	Each step in the process of gathering input to displaying output adds delay, with example amounts of delay below each step. Note the high 200ms latency from the monitor.	94

LIST OF ABBREVIATIONS

FPS	First-person shooter videogame
RTS	Real-time strategy game
FPS	First-person shooter game
RPG	Role-playing game
TTR	Time to react
HCI	Human computer interaction
VR	Virtual reality
AR	Augmented reality
JND	Just noticeable difference

CHAPTER 1

INTRODUCTION

1.1 Motivation

Video games are a major type of entertainment for millions of people. In 2018, the videogame industry made over 138 billion USD in revenue, which is nearly three times as much revenue as both music and film industries combined [47]. Newzoo’s 2018 videogame industry report claims this massive amount of revenue comes from more than 2.3 billion gamers [57], showing that an enormous amount of people both care about and pay for games. For game designers it is thus financially important to design the best games they can.

Many of the most popular videogames and game genres involve fast reactions. First-person shooters such as APEX: Legends and Fortnite, fighting games such as Street Fighter and Mortal Kombat, multiplayer online battle arenas (MOBAs) such as League of Legends and DOTA, infinite runners such as Geometry Dash and Temple Run and real-time strategy games such as Starcraft all require fast reactions. Fast-paced games such as CounterStrike, Smash Bros. and Starcraft are especially favoured in the E-sports scene, which is predicted by 2020 to surpass traditional sports such as soccer or football both in terms of popularity and revenue [22].

It is difficult to quantify what makes a ‘great’ game, or what will make a game popular, but it is widely acknowledged that having ‘sluggish’ or ‘sticky’ controls is undesirable. It is critical for player performance and experience that the game is responsive. Responsiveness means how quickly the game responds to input. How responsive a game needs to be is game and genre-dependent. Games on slower timescales do not need to be as responsive as games requiring split-second decision making.

One of the major contributing factors that can make games less responsive is local latency. Local latency is the total delay between input and its resulting output to the screen. This latency is produced by a combination of delays including polling input from input devices, processing delays from the software, and processing and refresh latency in the display device. A survey of real-world gaming setups found local latency to vary from 23ms up to 243ms, which is noticeable by players

and may significantly degrade player performance and experience [39].

Local latency is different from other forms of latency that have been studied in gaming contexts. Network latency is one such well-known example, and is the delay between transmitting information and receiving it between computers over a network. Network latency only affects networked applications, whilst local latency affects both networked and non-networked applications. In terms of severity, local latency has been found to often exceed that of network latency [66].

In terms of actual effects, local latency can substantially reduce player performance and player experience in fast-reaction games, leading to frustration and poor retention of audience. Even small amounts of local latency can significantly affect experience and performance [1, 42]. Due to these noticeable effects, specialized ‘gaming’ hardware such as gaming mice or 144Hz monitors are available to reduce local latency as much as possible. The popularity of this specialized gaming hardware shows that gamers are aware of and care about local latency.

It is therefore important for game designers to understand what local latency will do to a game in order to try and make the game resilient to local latency. Input device, task, game speed, and almost every aspect of a game factors into how sensitive a game is to latency. The aforementioned factors can be designed to make a game more resistant to latency, but we must first understand how each individual factor is affected by latency.

1.2 Problem

The problem is that there is currently little knowledge about how local latency affects games. There have been relatively few studies that examine latency in games, which typically fall into one of two categories: bottom-up studies or top-down studies. Both study types have limitations, and on their own do not provide enough information on how games are affected by latency.

Bottom-up studies examine a specific game to see how it is affected by latency [4, 74]; however, gameplay mechanics can differ greatly even between games of the same genre. Top-down studies examine simple tasks, such as pointing [34, 53], without a surrounding theme or complications of other gameplay elements. However, in real games it can be difficult to separate tasks from others. Lacking this insight, designers and game companies carry out time-consuming play-testing with multiple levels of latency for each game scenario or task.

If more knowledge on local latency and its effects was available, game designers would be able to design more latency resistant games to improve their responsiveness, and reduce time-consuming

latency-related play-testing. In this thesis, we address this problem through three studies to gather knowledge about how local latency affects different input devices and game atoms, and how it interacts with game speed. We also provide a predictive model that can quickly identify and quantify latency-sensitive elements in games.

1.3 Solution

Our solution is to provide game designers with information about the effects of local latency on games and to generalize this knowledge into a predictive model. This will both allow designers to design better latency-resistant games, and reduce needed latency-related play-testing. Pertinent information on latency’s effects includes information about both player performance and player experience. Games are meant to create experiences, thus making player experience just as important to study as gameplay performance. This solution comes in three parts: how latency affects performance and player experience with different input devices, how latency and interaction speed affect gameplay atoms and player experience, and lastly predicting latency’s effects.

First, we must examine how games are played by examining various input devices. There are a plethora of different input devices to play games including mouse, controller, touchscreen, stylus, AR, VR, Wii-mote and Kinect. Each device has different characteristics such as direct/indirect input and absolute/relative movement. These different characteristics increase the likelihood that each device is affected by latency differently, and that each device will be differently suited to various tasks. Game designers must carefully choose input devices that are both latency-resistant and suitable to core gameplay tasks, and then design their game around these devices. Unfortunately there are relatively few cross-device studies examining the effects of latency.

To fill this gap in knowledge, we conducted a study comparing pointing performance and player experience across four different devices. Chosen devices covered a broad range of input characteristics, and were representative of common gaming platforms as well. Both stationary and moving target acquisition tasks were performed and analyzed. Task cursor paths were also analyzed to make a number of design suggestions and recommendations for different types of pointing devices.

Second, we must examine how latency and interaction speed affect gameplay ‘atoms’. Each game is composed of a combination of simple tasks or atoms including targeting, tracking, interception, dragging, and sequences of key presses. These interactions are often combined or performed simultaneously to create the more complex gameplay seen in today’s games (e.g. simultaneously

aiming to shoot an opponent whilst moving to dodge incoming damage in a shooter).

In addition, the difficulty of these tasks is linked to interaction speed — the speed at which the player must react in order to successfully complete the gameplay atom. Previous work also indicates faster interactions are more sensitive to latency, demonstrating game speed to be an essential element in understanding latency’s effects.

We carried out a study that examined the relationships between interaction speed, local latency, and game performance. The study featured a simple task — target interception whilst playing a Pong-like game. Both latency levels and interaction speed were varied in order to determine effects on performance and player experience.

Last, we can extend our understanding to predicting latency’s effects by building a predictive model. Game companies don’t always have time to perform large-scale user studies on their game, thus making any time saved economically beneficial. A predictive model would reduce play-testing, as multiple latencies could be modelled ahead of time and a reasonable game speed chosen for critical gameplay sections. Although latency can cause a wide range of negative effects such as poor player experience or a declining user base, we focused on user errors such as failing to complete time-sensitive tasks, or misclicking due to latency. If error rates due to latency can be predicted, sensitive and problematic gameplay tasks could be identified.

Using data from the Pong study we developed a model for predicting error rates given game speed, latency, and number of simultaneous gameplay tasks. We outline our method and steps so game designers can easily find predicted error rates for their specific atom of interaction. This generalized model is useful because it can be applied to any game, including games featuring multiple simultaneous tasks, such as dodging and shooting. Multiple game speeds can be tested to see which would be problematic under normal amounts of latency.

1.4 Steps to a Solution

There were many steps involved in accomplishing our goal of providing relevant information about latency’s effect on games. Since our solution comes in three parts, we divide this section in three as well.

1.4.1 Effects of Latency on Input Devices

- In order to perform an empirical study examining performance with various devices under latency conditions, a task had to be chosen. This task had to be common, simple and have its performance be easy to measure. Pointing and target interception were thus natural fits, as they are extremely common for both games and non-gaming applications. Fitts' law and its many variants are well-known and provide an easy-to-use way of measuring pointing performance [49]. There is also a substantial amount of literature on pointing to guide the design of the study.
- The next step to understanding how latency affects performance with different input devices was to decide which devices to examine. Since the first videogame was invented in 1958 [24], there have been a wide of variety of game platforms and input devices. For our work to be relevant today, input devices had to be representative of what the average gamer uses today, and thus had to be both popular and common. This list was then narrowed down to only pointing devices that fit with our tasks. Finally, pointing characteristics such as direct/indirect input and absolute/relative movement of each device were examined. Four devices were chosen to cover a broad range of these input characteristics (Figure 2.3). Chosen devices included mouse, gamepad, touchscreen and drawing tablet.
- Task parameters then had to be decided upon. For pointing and target interception, multiple values for distance to target, target size and target movement speed as well as latency all had to be chosen. Values were chosen to provide a range of difficulties (determined by testing with each input device). Added latency values ranged from 0-350ms.
- Player experience measures also had to be decided. After consulting a wide variety of scales, we found that no single scale captured all measures we wanted to study. We decided to use questions from a variety of scales to capture as many experiences in as short a time as possible. Questions were chosen from the following scales: Player Experience of Needs Satisfaction (PENS), Intrinsic Motivational Inventory (IMI), and Task Load Index (NASA-TLX) [69, 68, 28]. Player experience questions were administered periodically between experiment rounds, and administered on a 5-point Likert scale (from Strongly Disagree to Strongly Agree).
- We then ran participants through a study where each participant used each device for pointing and interception tasks. Performance and experience measures were recorded and analyzed.

Based on our results, we created a number of guidelines and suggestions for game designers when using certain input devices under latency conditions.

1.4.2 Effects of Latency and Game Speed on Game Atoms

- The first step in providing game designers with pertinent information on latency, game speed and game atoms is to choose which game atoms to measure. In order to generalize our results, chosen game atoms must be fast-paced and sensitive to latency whilst also being featured prominently in games. Chosen atoms must also have a measurable performance outcome, whilst being short and easily repeatable. Lastly, as there are many game atoms, we are interested in starting on the very simplest of atoms. Like the previous study, target interception fit this criteria. Traditionally, target interception is done in two dimensions (X and Y), as this translates well to our flat 2D monitors, however it can also be done on one dimension (X or Y). 1D interception was chosen due to its simplicity, and how it ties into the next step.
- Next we had to implement 1D target interception in a game setting. The closer the whole study is to an actual game, the more valid our results will be to similar types of games. We also wanted a game genre that most users will be familiar with, and is easily understood. After examining various game genres, one style of game featuring 1D interception stood out: ball-return style games such as Pong, and Breakout. We chose to implement our own single-player version of Pong, as it was both easy to implement, and had fewer distracting elements.
- With task decided upon, we had to choose an input device to use. Since we are interested in testing many different game speeds, we would not have time to test multiple input devices. The chosen device had to be both popular and a natural fit for the task. For 1D target interception, the following devices were considered: mouse, gamepad and a traditional paddle (used in arcade cabinets). The wheel was dismissed as not being relevant enough today to study, and the gamepad has traditionally poorer performance than the mouse. This left the mouse, which was chosen as the input device for the study.
- Parameters for the task then had to be chosen. Pong features a player-controlled paddle that must intercept a moving ball. To vary the game/interaction speed of the task, the movement speed of the ball was changed. Speed values were calculated as the time it took the ball to move from the top of the screen to past the player's movement axis. Reasonable game speeds

chosen from testing gave the user 400-600ms to react. Other parameters included ball/target size, player paddle size and added latency. Parameter values were chosen to provide a range of difficulties by having ball and paddle size remain constant, while adjusting the ball speed and added latency amounts (with added latencies ranging from 0-250ms).

- Player experience questions were then also determined. A wide variety of scales were selected from and used in a similar fashion as the Input Device study (subsection 1.4.1). Experience questions would be asked after each round of gameplay.
- The last step was to run the study, and analyze the error rates (number of times ball was missed) and player experience in conjunction with game speed and latency.

1.4.3 Modelling Error Rates

- The first step in building a model to predict latency's effects is determining what kind of model to use. After research, logistic models were found to represent human performance well. Logistic models (or psychometric functions) are often used for determining just-noticeable differences between stimuli of varying strengths. This function measures a number of binary observations (either the user noticed a difference in stimuli, or they did not) and produces a percentage likelihood of the user being correct. This lends itself well to binary error outcomes in games: either an error occurred, or it did not. This does not measure task completion time (another useful performance metric), but many completion time tasks can be rephrased using a binary outcome (e.g. needing to click on the target within two seconds).
- The next step was to determine what measures should be included in the model. Obviously latency needed to be included, but based on the previous effects of latency and game speed on game atoms study (subsection 1.4.2), we knew that game speed was critical in determining a task's sensitivity to latency. Since game speed affects how sensitive a task is to latency, we combined both game speed and latency into a single measure called Time to React (TTR). As the name implies, TTR is how much time the user has before they must react (or an error will occur). TTR will be used as the main variable in the logistic equation.
- Other parameters such as floor, ceiling, curve midpoint and steepness also had to be chosen in order to use a logistic equation. Using data from the effects of latency and game speed on game atoms study (subsection 1.4.2), we chose the ceiling to be the expected errors from

chance (or errors if the user does nothing), and the other parameters to be determined by the number of simultaneous tasks being completed. In the effects of latency and game speed on game atoms study the only task was intercepting the ball. In other games the player must often perform several tasks at once, such as dodging attacks whilst shooting. It is probable that the more interaction effects/tasks there are, performance will degrade.

- Lastly, additional validation was needed for the model. We needed a slightly more complex game involving multiple tasks/interaction effects, bringing us closer to more complex games played today, yet still be able to compare results to the previous effects of latency and game speed on game atoms study. We performed another study using a game based on Space Invaders, with 1D pointing as the task. The study featured two simultaneous tasks: dodging enemy bullets and shooting enemies. Task, input device and parameters were decided upon in a similar fashion to the Pong study.

1.5 Evaluation

In this section we evaluate whether our solutions of performing studies and creating a model allows game designers to make better games. This goal is difficult to evaluate, and cannot be evaluated empirically. Instead we focus our evaluation on analytically and theoretically discussing if our newfound knowledge and model can reduce needed play-testing related to latency, which is discussed more in the Discussion chapter. Lastly, some of our evaluation, such as validating our model, is simultaneously a contribution. We now split this evaluation into three sections mirroring our three solution sections.

1.5.1 Effects of Latency on Input Devices

To test the effects of latency on input devices, participants used four different input devices for both stationary and moving target acquisition tasks (seen in chapter 3). Devices included mouse, gamepad, touchscreen and drawing tablet. Before conducting the study, we ran in-lab tests to determine reasonable latency amounts to add to each task. By varying target widths and target distances, we measured Fitts' Index of Performance (IP or throughput) while also recording cursor paths. By using an established metric such as IP as our main performance metric, we determined an ordering of devices best suited to our tasks, while also determining their sensitivities to latency. Cursor paths were used to measure cursor path metrics described by Mackenzie [51], and were used

to explain latency’s exact effect on different devices.

For performance, we found the touchscreen to perform the best on both stationary and moving targets, the mouse to overall perform well, and both drawing tablet and controller to perform poorly. For player experience, the touchscreen was rated most positively whilst the other devices were grouped together with lower ratings. We also found each device to have unique cursor paths and path metrics, with the mouse having accurate paths, the controller having straight but inefficient paths, and the drawing tablet having highly erratic paths and difficulty tapping the target. We determined the most important path metrics to be target re-entry (TRE) and movement error. Based on the above results we also provide device-specific design recommendations for device suitability and pointing under latency conditions.

Based on the newly gathered information and guidelines, game designers can now make informed choices for choosing a latency-resistant pointing device based on empirical evidence. Additionally, input device designers can refer to our results to know which cursor path metrics are most correlated with performance. Lastly, the provided guidelines allow designers to design more latency-resistant interactions for a chosen input device.

1.5.2 Effects of Latency and Interaction Speed on Game Atoms

For determining latency’s and game speed’s effect on both player performance and experience on game atoms, we performed a study based on a Pong-like game. The study featured moving target interception with a mouse, and varied both game speed and added latency levels. Performance was measured as a normalized error rate between 0 and 1, where meant 0 no errors occurred and 1 meant the player missed the ball each time. Player experience from a variety of scales was also collected between each round.

Our results confirm that game speed and latency are linked, and that game speed determines how sensitive a task is to latency. By combining both game speed and latency, we create a new measure called Time to React that is useful for analyzing results from multiple game speeds simultaneously. By examining Time to React we also determine latency’s effects are not simply linear, and thus create a modified Time to React measure. Lastly, player experience was found to be significantly and negatively correlated with both latency and game speed.

For game designers, Time to React offers a promising new way for examining play-testing data combined with latency and game speed. We also confirmed that game speed affects latency sensitivity and that latency’s effects are exponential on error rates for the first half of the sigmoid curves,

thus providing further proof that latency needs to be taken seriously, as even 100ms of latency can seriously affected moderately fast-speed interactions.

1.5.3 Modelling Error Rates

To create a predictive model of error rates from game speed and latency, we used data from the previous effects of latency and game speed on game atoms study (subsection 1.4.2). Using our new Time to React measure, we could see performance curves of different game speeds converge towards a single performance curve. Using a modified logistic model, we estimated reasonable values for other parameters, knowing that our model had to have a parameter for multiple simultaneous tasks/distractors.

To evaluate our predictive model, we ran a narrow verification test with inputs similar to what the model was trained on. The validation study was based on a Space Invaders-like game using a mouse, and involved two simultaneous tasks: dodging enemy bullets and shooting enemies. Like in Pong, both game speed and latency values were varied to analyze a normalized error rate. The similarities in task and parameters allow for the comparison of data between studies, and tests our model on a slightly more complex game with generally slower game speeds and two tasks instead of one.

Our results had our model predict error rates with reasonable accuracy. For using our model we provide game designers step-by-step instructions as well as how to choose a game atom to test. With this model in hand, substantial latency-related game testing per scenario/task can be saved. Multiple game speeds and latencies can be simultaneously compared to immediately identify problematic and latency-sensitive gameplay sections, as well as determine reasonable game speeds that are resistant to latency.

1.6 Contribution

This thesis provides game designers with missing knowledge related to local latency’s effect on games, allowing designers to both design better games and reduce latency-related play-testing. This contribution comes in three sections.

- First we inform designers how common gaming input devices are affected by latency for two common tasks: pointing and interception. Pointing is ubiquitous when using a computer, and is common in many game genres such as RTS, and RPG. Interception appears almost

exclusively in games, and is very sensitive to even small amounts of latency due to its time-sensitive nature and how the target can move out from beneath the cursor. If interception is a core mechanic for your game, it is imperative to know both which device is most resistant to latency and which devices performs best. We provide a cross-device comparison of pointing and interception performance which designers can use when selecting an input device for their game. We confirmed that input devices are indeed affected by latency differently, and found some surprising results (touchscreen is still the best for interception, but is affected by latency) and analyze which cursor path characteristics are responsible. Unlike most studies, we provide cursor path visualizations, which provide meaning to otherwise difficult-to-interpret numeric cursor path metrics. The two most important cursor path characteristics are identified, thus placing an emphasis on what should be prioritized when testing or judging new input devices. Lastly we provide six design recommendations for specific devices that are immediately useful and usable by game designers.

- Our second major contribution provides information on how one of the simplest game atoms, 1D target interception, is affected by both game speed and latency. Each game is composed of a number of simple interaction atoms. Since these atoms are often repeated many times during the core gameplay loop, it is imperative to understand how to both identify these atoms and understand how they are affected by latency. We start by providing guidelines on how game designers might identify game atoms within games. Our study confirms that latency's effects are exponential, rather than linear. We also examine game speed, which changes how sensitive a task is to latency. Since we confirm that game speed and latency are linked, we create a new measure called Time to React in order to help analyze latency's effects. Time to React allows for easier comparisons of performance data featuring multiple game speeds and different latency values.
- Our final major contribution is modelling or predicting error rates for a given game atom. Using Time to React, we created a predictive model that predicts error rates given latency and a game speed. Error rates are immediately useful to game designers, as high error rates will result in frustrated players. Our model also allows for performing multiple tasks simultaneously, allowing it to be used on more complex games. We link Time to React (game speed and latency) to errors, and explicitly outline steps on how to use this model. The model is simple to use, and can be used on single or multiple game atoms at a time. Game companies spend

considerable amounts of money play testing games, and our model will reduce the amount of needed play testing by estimating error rates at different levels of latency. We hope further testing and validation of our model will occur on more interaction atoms in the future.

Lastly we list a number of additional minor contributions:

- For the input device study, controls and calibration for four different pointing devices had to be performed. The indirect input and relative movement of the gamepad required much testing to calibrate proper cursor speeds and thumbstick dead zones.
- We adapted both Pong and Space Invaders to be suitable for studies. We provide information on these reference implementations for future use and study.
- We also identified a number of different game atoms (though we did not study all of them).
- We provide a useful link between psychometric/logistic functions used in psychology to determine just-noticeable differences between stimuli and binary error rates from player performance in videogames. It is worth investigating models or methods from other fields to see if they can be applied in regards to latency.

1.7 Thesis Outline

Chapter Two provides a literature review of related work of games research, and HCI. This includes surveying work on latency, how latency affects specific games and genres, input devices characteristics, models of human performance such as Fitts' law, and lastly player experience scales such as Intrinsic Motivation Theory (IMI) or Player Experience of Need Satisfaction (PENS).

Chapter Three presents the first of three studies, which evaluates various pointing devices in pointing and interception tasks under latency conditions. We describe participant recruitment, study apparatus and our methods. Performance measures included standard Fitts' Index of Performance, cursor path metrics and player experience. The results from this study showcase significant performance and path differences between devices.

Chapter Four presents the second study, which examines how latency and game speed affects game atoms. Specifically, we examine target interception with a mouse on a custom singleplayer-only version of Pong. We first provide guidelines on how to identify game atoms, and then identify a number ourselves. We describe our participant recruitment procedures, study apparatus and methods. Both performance and player experience is analyzed, along with their relationships between

latency and game speed. We finish by presenting a new measure combining both game speed and latency called Time to React.

Chapter Five presents a predictive model based on the results of the study in Chapter Four. We describe the steps used to build the model, then discuss how it applied to the study data from Chapter Four. Lastly we outline the steps game designers would follow to apply the model.

Chapter Six presents the third and final study, which is validation for the predictive model described in Chapter Five. This study is similar in design as the study in Chapter Four, but features a more complex task involving multi-tasking in the form of simultaneous dodging and shooting in a custom version of Space Invaders. Results and analysis of the study are presented, along with an analysis on goodness of fit of the model.

Chapter Seven discusses more generalizable results and insights from our studies. Real-world uses of our insights and models are discussed, and design guidelines for various input devices are presented. Findings on game atoms and game speed are also discussed, along with the applications of our predictive model.

Chapter Eight concludes the thesis with a summary of our work and contributions.

CHAPTER 2

RELATED WORK

To provide game designers information on how latency affects both input devices and game atoms, three main areas of research must be examined. First we begin with latency and its various types, and then move onto game input devices, and finish with the relationship between games and performance.

2.1 Latency and Games

Latency in computing simply means a time delay. There are several types of latency; input lag, display lag and network lag. Latency has been shown to negatively affect many computing applications including digital sketching [5], collaborative groupware [75], video scrubbing [54], motion tracking [62], as well as many networked applications [75, 48, 4].

Video games have proven to be especially vulnerable to latency [4, 39, 74, 20, 12], since they often require quick and precise movements that have low or urgent deadlines. Deadline, as defined by Claypool, is the time an action must be completed by [12]. Especially vulnerable game genres include shooters, fighting games, rhythm games, racing games and fast-paced real-time strategy games. Studies of specific games include the popular shooter series Unreal Tournament and Counterstrike [4, 66, 20], the popular RTS game Warcraft 3 [74, 20], as well as some cross-game comparisons [66].

These video game studies report a wide range of latency thresholds, where latency begins to significantly degrade player performance. One racing game study saw performance begin to degrade at 150ms of latency [61], and other FPS games found latencies above 100ms [4] and 150ms [20] to affect performance. Online role playing games appear to be more tolerant of latencies, and are resistant up to 150ms-250ms of network latency[20]. Claypool did a survey that suggests games with a third-person camera can perform well until latency reaches about 500ms [12].

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 [12]. 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.

We will now review two specific forms of latency that are most encountered in games: local latency and network latency.

2.1.1 Local Latency

Local latency (sometimes called input latency) is the total elapsed time for gathering user input and displaying the resulting output to the screen. Unlike network latency, local latency affects both singleplayer and multiplayer games. Since there will always be some amount of local latency, all computing applications are subject to local latency. Below we describe a number of sources that can add to local latency during the process of capturing input, to displaying a user's input (Figure 2.1).

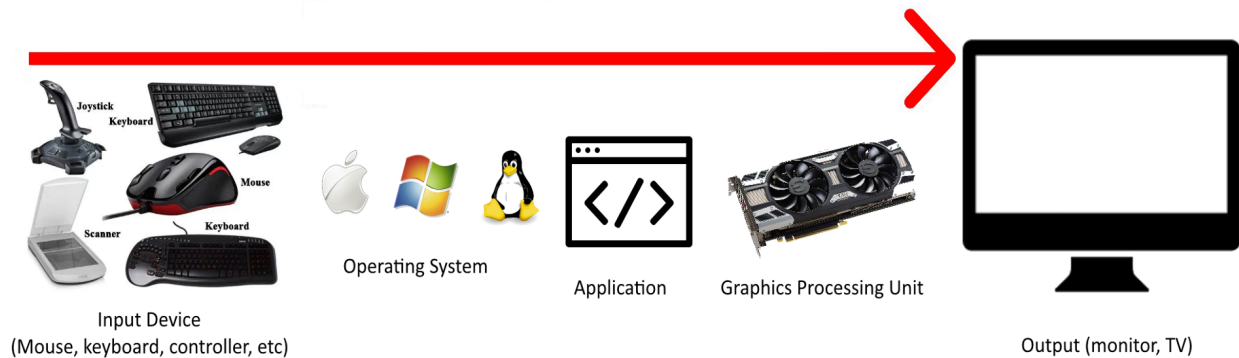


Figure 2.1: Each step in the process of gathering input to displaying output adds delay known as local latency.

1. **Input Devices:** An input device is required to send input to the computer. The device is periodically polled for new input. The polling rate is generally measured in Hz (cycles per second). To calculate the total delay in polling input, Hz can be inverted to measure the seconds per cycle. Input delay values from common gaming input devices can be found in subsection 2.2.2. Generally, input devices take 1-10ms to report a new input.

2. **Operating System:** The signal data from input devices is then passed into a device controller. The device controller acts as a bridge between the input device and the operating system. Each device controller temporarily stores received data on its own buffers and data registers. Each device controller also has an associated software program called a ‘device driver’ that is used to communicate between device controller and operating system. Depending on the input device, device drivers sometimes perform filtering or smoothing. Once input has finally passed through the device controller and device drivers, the operating system then passes the input onto the correct software application.
3. **Software applications:** Games operate in discrete frames which are commonly updated at a rate of 30 or 60 frames per second (33ms/frame and 17ms/frame). This frame rate determines the amount of time that elapses between frames, and thus how long it will take to display new information. Additionally, sometimes input or output buffering is performed, which may add 67ms of delay or more [39].
4. **Graphics Processing Unit (GPU):** To ease the load on the CPU, graphics cards are used to perform computations related to displaying on-screen graphics, such as those found in games. Software applications send information regarding the scene needing to be rendered such as 3D geometry, camera position, and lighting to the GPU for processing. The time taken to render a scene is dependent on the quality and complexity — if the scene being rendered is too complex, the render rate of the GPU will decrease, thus causing additional latency. When finished, pixel colour values will finally be sent to the monitor to be displayed. The render rate of the GPU is dependent on the frame rate of the game, and the refresh rate of the monitor.
5. **Monitor:** Lastly, all monitors have a maximum refresh rate dictating how fast new information can be displayed. Although older cathode ray tube (CRT) monitors have adjustable refresh rates, LCD screens commonly have a 60Hz refresh rate (33ms between updates), although gaming LCD monitors can have refresh rates of 120Hz or 144Hz (8 and 7ms). Game frame rates and GPUs are often synced to the monitor’s refresh rate (called vsync), as full screen updates cannot be shown any quicker than the refresh rate. On-screen pixels also take time to change colour, with times ranging from 1-12ms [3, 2]. Lastly, some monitors/televisions perform post-processing effects such as colour adjustment and motion smoothing. This can add significant amounts of latency, with some TVs from 2018 adding over 100ms of extra

latency [38].

As the above steps show, there can be large differences in local latency between real-world gaming setups. In 2015 Ivkovic et al. found real-world gaming setups to have between 23ms and 243ms of local latency [39]. These local latency differences between gaming setups is one reason why competitive Super Smash Bros. tournaments are typically played on CRT televisions that features higher refresh rates and no post-processing [67]. Beyond refresh rate, LCD monitors take time to change individual pixel colours (pixel response time), whereas CRT's can do this instantly. Additionally, all CRT televisions have a fixed amount of output delay, whereas LCD monitors each have different amounts of output delay due to post-processing. This is why the iconic NES light gun for the game Duck Hunt works on CRT monitors, but not on LCD monitors [35].

Although higher amounts of local latency poses significant problems, even small amounts of local latency can be problematic depending on task and setup. Anderson et al. conducted a study examining common touchscreen tablet tasks such as web page browsing, photo viewing, and ebook reading, and found direct-touch touchscreens to have a just-noticeable difference (JND) of 2ms of latency [1]. It should be noted that JND does not necessarily affect performance, but likely affects player experience.

A different study by Jota et al. used a custom setup with a projector to achieve and test low latency values not achievable by normal hardware [42]. They tested 1, 10, 25 and 50ms of local latency on direct-touch dragging tasks. They found latencies of 25ms and above to significantly and negatively affect dragging task performance, but noted that for simpler tapping tasks 50ms of latency made very little difference.

The above studies demonstrate that local latency poses problems in a variety of contexts, however local latency is not the only form of latency. Network latency is also commonly encountered (and commonly complained about), however Raaen et al. found local latency can be as large or even larger than regular network latency [66]. Raaen's study examined used cameras and oscilloscopes to determine local latency values of computers brought in by participants. Raaen concluded that local latency constitutes such a large share of the total delay that it should be studied when examining latency and games.

2.1.2 Network Latency

Network latency is the delay from sending a signal from one machine to another over a network, and has received substantial academic attention [9, 13, 14, 45, 9, 75]. Networked applications experience

a number of distinct challenges that make network latency and its effects differ from that of local latency.

Network latency is rarely constant. This variance in latency values is called jitter. Jitter causes problems for interpretation of smooth motion and streams, whereas latency causes problem for coordination of shared access to artifacts [25]. Methods for reducing jitter are listed in subsection 2.1.3.

Networks themselves can also be unreliable. Information does not always reach its destination over a congested network. This phenomenon is called packet loss. Transmission Control Protocol (TCP) detects packet loss, however it takes precious time to re-transmit the same lost information, often significantly increasing network latency. Packet loss can sometimes cause packets to arrive out of order, and cause 'jerkiness' or 'warping' of continuously tracked objects, such as shared cursors in networked groupware.

Network latency poses problems for various types of network-reliant applications, such as group collaboration tools like Google Docs and remotely operated surgical robots [75, 48]. Networked videogames are especially sensitive to network latency due to the often faster-paced nature of gaming. Networked multiplayer games are incredibly popular, and are always subject to some amount of network latency.

There have been many studies on networked games [4, 74, 20, 10, 71, 70, 59]. Network latency has been found to negatively affect both experience and performance in FPS games such as Unreal Tournament and Counterstrike [4, 66, 20], RTS games such as Warcraft 3 [74, 10], and sports games such as Madden NFL football [59]. As with local latency, the faster-paced the game and closer the player perspective is to the character (first-person view versus third-person view), the more sensitive the game is to latency. A study by Henderson et al. found that varying the latency on several servers of popular FPS games decreased the number of players joining the server, but did not cause players to quit a server they were already playing on [29].

One special case of combining both local latency and network is that of cloud gaming. A client sends input over the network to a machine that is actually running the game, and visual output is then streamed back to the client (seen in Figure 2.2). By using cloud gaming, even traditionally singleplayer games are subject to network latency. Additionally, both host and client machines have their own amount of local latency for sending input, processing and rendering the game, and displaying the output to the user's monitor.

Lee et al. found that different games and genres are more or less suitable to cloud gaming, and that FPS games are especially sensitive to network delays often found in cloud gaming [45].

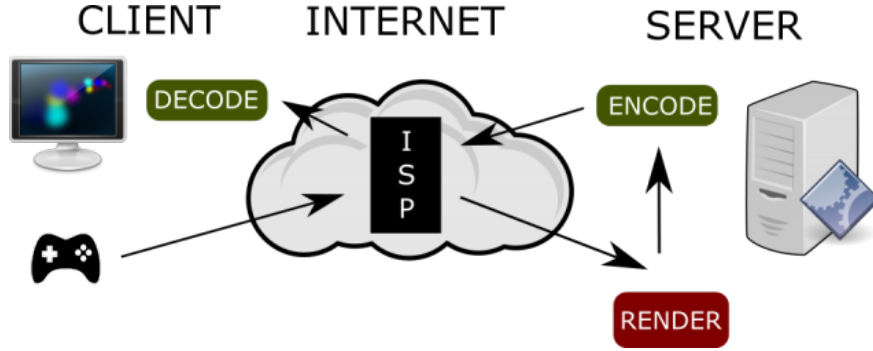


Figure 2.2: A typical cloud gaming setup. At the left side the user sends input over the internet to a server in the cloud. On the server the game scene is then rendered, and to save bandwidth, encoded. The encoded video is then sent back over the internet to be decoded by the user’s client [15].

Claypool performed a study comparing the commercial cloud gaming service OnLive, and the open-source service GamingAnywhere, and found that experience and performance degraded linearly with latency, and that cloud-based games are as sensitive to latency as non-networked shooters [9].

Popular and current cloud gaming services include Playstation Now, GeForce Now, LiquidSky and Microsoft Azure. Additionally, Playstation 4, Xbox One and Steam allow for the broadcasting and streaming of games from your console to be played on another device. However, the announcement of Google’s cloud-gaming platform Stadia may indicate cloud gaming will soon be on a meteoric rise. One research firm estimated that consumer spending on cloud gaming subscriptions reached \$234 million USD in 2018, and is forecast to rise up to \$1.5 billion by 2023 [16].

In terms of specific gameplay problems caused by network latency, one fairly typical problem particularly relevant to shooters is a player’s true location differing from where they appear on-screen [12, 20]. This causes different players to have different views of what is happening, and it can be visually disruptive when this discrepancy is repaired. These symptoms are due to the local client’s game state differing from that of other players.

Savery et al. report a number of shared entities must be synchronized across a network to maintain consistent states between local and remote clients. However each class of entities has different consistency requirements. Consistency requirements are determined by whether or not entities can be interacted with by single or multiple players, whether the entity affects critical game variables and how often it is interacted with [71]. Entities can diverge in consistency in three ways: magnitude (state), time, and rate of change. When inconsistencies between local and remote game states appear, they must be repaired. They can be repaired either: immediately, smoothly over

time, or not repaired at all. Choosing the correct repair method significantly affects both player experience and performance [71].

2.1.3 Methods for Combating Latency

The simplest way to combat latency is to learn to play with higher levels of latency. Humans can accommodate some latency if movement is predictable [63] or delays are visualized [26], however performance loss will likely still occur. In terms of player experience, players are capable of noticing latencies as low as 2ms for certain tasks [1]. This indicates even if latency could be accommodated by learning, it is likely they will still notice the latency and their experience will still suffer.

Other methods exist to reduce performance loss due to latency [70, 71], however most have significant trade-offs. Some are only employed for combating network latency, while others can be employed to combat both local and network latency. Below we list several types of methods for reducing latency’s effects.

Prediction methods

Prediction methods rely on being able to predict future player actions while waiting for actual player input. Dead reckoning is one such algorithm, and is commonly used in networked games. It assumes objects will continue moving (or not moving) in their previous heading and is used to reduce unnecessary network updates [80]. This works well if player movement is not erratic, and players continue doing what they were doing previously.

If the prediction is incorrect, the game state must change to what it ought to be. This repairing of game states can cause visual discrepancies such as your character abruptly teleporting somewhere, or your character dying to unseen projectiles or attackers, and can be very disruptive [71]. Designers must carefully consider how consistency repairs will make players feel: abruptly dying to unseen fire feels completely unfair, and may ruin a player’s experience if done too often.

Aim Assist

Some techniques can be applied to help specific tasks in light of either network or local latency. Targeting, or aiming, is an important task in many game genres, and due to high-precision and often urgent time requirements, one that is highly sensitively to latency [8]. Aim assist, target magnetism, or sticky targeting is a technique used in aiming or targeting tasks to increase targeting performance, move the cursor onto a target more quickly, or allow the cursor to remain on a target

more easily. One study applied high amounts of latency to targeting in a first-person shooter, and found aim assist to drastically increase targeting performance [39].

There are several ways to implement aim assist. One method is via ‘magnetism’, where if the cursor is close enough to the target it is automatically ‘pulled’ towards it. Another method is via ‘sticky’ targets, where cursor movement sensitivity is altered when moving towards or away from the target, making your cursor ‘stick’ to the target.

Input Delay

Input delay (sometimes called local lag) is a consistency maintenance method for networked applications that counter-intuitively increases overall latency to improve player experience and performance [71, 70]. Input is sent across the network as usual, but is not acted upon until a certain amount of time elapses. This method helps the consistency and experience of players with slower internet speeds and high packet loss by allowing for late network packet arrivals. This method helps overcome the effects of network latency by changing the network latency into essentially a larger pre-determined amount of local latency. This minimizes the effects of slow or congested networks, and provides a more uniform experience to all players.

An in-game example of networked input delay would be pressing the A key to move left, however your character does not move left until after the delay has passed. Developers must choose a delay constant to balance between input responsiveness and late network update arrivals. Local lag increases total latency, but reduces inconsistencies between client game states.

Remote Lag

Remote lag is a consistency maintenance method for networked applications that is based on a similar idea as that of local lag. Remote lag buffers multiple network updates before updates are used, and allows for receiving updates out of order. Most commonly used in streaming applications for video or audio, it can also be used in games. Having multiple updates available allows for smoothly interpolating between them, reducing any jarring or jerkiness due to delay between network updates. This increases the total amount of latency, but improves visual consistency of the game and may improve player experience [71, 70].

Game Design

The above methods may help reduce to reduce latency’s effects, and increase both player performance and experience, however games and applications should still be designed to be as latency resistant as possible. To this end, we must learn how to determine if a game is sensitive or not to latency. This in part depends on the input device(s) being used – some devices may be more or less resistant to latency for given tasks.

2.2 Input Devices

All computer-human interactions require an input device. As described by Hinckley et al., input devices sense physical properties of people, places, or things [33]. Input devices transform physical changes from the user into electronic input. These devices can take various forms, and use various sensors. In this section we describe previous work done on input devices and latency. We start by describing pointing device properties, and end by describing research about popular gaming devices and latency.

2.2.1 Pointing Device Characteristics

Although there are many input devices, pointing devices are amongst the most prominent types of input devices. Pointing devices are used to control a virtual cursor or targeting reticle. The act of moving a cursor to a target is called pointing, targeting or aiming. It is instead called target interception (or simply interception) if the target is moving. This can be done in 1D, 2D space or 3D space. 2D targeting is far more common due to being used in almost all desktop/laptop operating systems. Since pointing is a fundamental task, it is important to understand how pointing devices differ. Below we describe various pointing device properties identified by Mackenzie et al. [50].

Device Resistance

Some devices require resistance (or lack thereof) to measure physical change. This determines whether or not the device is in a fixed position, or can be moved. There are three categories for device resistance:

- **Isotonic** devices are movable, and measure displacement. Examples include moving the mouse to move the cursor.

- **Isometric** devices measure the force exerted upon them. Examples include trackpoints on laptops (red nub pictured in bottom right of Figure 2.3), and force-sensitive touchscreens.
- **Elastic** devices have increasing resistance to force applied. Examples include arcade joysticks, where the more you push against them the more they push back.

Property Sensed

The type of physical change sensed by the device. Below we list the more common physical properties that are measured [52]:

- Position
- Motion
- Force
- Angle (for rotary devices)
- Change in angle (for rotary devices)
- Torque (for rotary devices)

Input: Direct/Indirect

Whether visual output is displaced from input space.

- **Direct input** devices do not require an intermediary between the user and display, since the display space is also used to enter input. Examples include touchscreens, where the user is directly tapping on the screen.
- **Indirect input** devices do not have their output and input spaces shared. Instead the indirect input device acts as an intermediary between the physical change being measured and the resulting output. Examples of indirect input devices include gamepads and the mouse.

Movement: Absolute/Relative

The mapping between points in input space and output space.

- **Absolute movement** devices provide a consistent mapping between points in input space and output space. For example, a user knows that tapping on the corner of a touchscreen will always result in a corner tap. This ensures that a given input (tap) always has the same given output (tapping at that location).





	Direct Input	Indirect Input
Absolute Movement		
Relative Movement		

Figure 2.3: Input devices, sorted by properties. Top left: smartphone with touchscreen, stylus and touchscreen with Nintendo 3DS, touchscreen of Microsoft Surface Studio. Top right: drawing tablet (absolute mode), WiiMote. Bottom right: laptop pointing stick, laptop track pad, thumbsticks on gamepad, mouse, joystick. Bottom left: soft joysticks on touchscreen.

- **Relative movement** devices provide an inconsistent mapping between points in input and output space. Using a desktop interface as an example, moving the mouse to the left moves the cursor to the left *relative to its current position*. The overall result of the input is dependent on the current state of the system, and does not guarantee that a given input (moving the mouse to the left) always has the same output or result (moving the cursor to the left to hover a specific icon).

Rate-control

All devices have a control-to-display (C/D) ratio for converting physical measurements into digital displacement values. Mice and other devices however can have different settings to change the ratio of physical displacement to virtual cursor displacement. For the mouse, dots per inch (DPI) is a measure that means moving the mouse one inch moves the cursor some number of pixels. For example, moving the mouse one inch at 800 DPI moves the cursor 800 pixels. The sensitivity of many pointing devices can be adjusted in various settings menus.

Depending on the resistance type of the device, certain devices can be limited in how quickly the pointer can be moved. Examples include isometric or elastic thumbsticks and joysticks, which have a device-limited maximum speed such as pressing as the thumbstick as far you can to a side.

Other devices, such as the mouse, can measure physical change fast enough that it practically has no upper limit to its control/movement speed; it moves as fast as your hand can.

2.2.2 Input Devices and Latency

In addition to each pointing input device having unique properties, each device also has an inherent amount of input latency (sometimes called input lag - not to be confused with the consistency maintenance method of input lag). This is known as the polling rate, and it is the amount of time taken for a computer to receive new input from the input device. Below we list a number of common pointing devices.

- **Mouse:** Most consumer mice poll at 125Hz, giving 8ms of delay. Gaming mice normally have an adjustable polling rate, going from 125Hz-1000Hz (1-8ms).
- **Touchscreens:** Latency for touchscreens is the delay between a tap occurring, and it being reported to the operating system. Unfortunately many manufacturers do not report touch latency. The touchscreen used in one of our studies (Microsoft Surface Studio with 28" PixelSense display) reports 8ms of touch latency.
- **Gamepads:** Again, manufacturers often do not directly report latency values. PS4 controllers plugged into Windows are found to have 2ms of latency [18], while we found a USB Xbox 360 to have 8ms (see chapter 3).
- **Drawing Tablet:** Drawing tablets commonly have 8ms of latency [76].

As noted above, pointing devices all perform the same task of controlling a virtual cursor (except for touchscreens), but they can vary greatly in how they function. Therefore, they are likely to be affected by latency differently. Below we present past work on the relationship between latency and common gaming pointing devices.

Mouse

The mouse measures displacement for its relative movement, and has indirect input. Mouse cursor speed is not rate-restricted, as the cursor can be moved as fast as your hand can move. The mouse has managed to become ubiquitous despite being somewhat counter-intuitive; it requires users to learn the relationship between moving their hand (and mouse) and seeing the virtual cursor move. Using a mouse has become second-nature to many who use computers much of the day, however

learning how to use the mouse is far less intuitive than learning how to use a direct input absolute movement device such as a touchscreen. Despite that hurdle, the mouse is the default device for PC gaming (though gamepads are becoming increasingly popular).



Figure 2.4: MSI Interceptor DS B1 gaming mouse.

In terms of research, the mouse is well-studied. It is a benchmark for pointing performance, and has been used to verify many adaptations of the famous pointing performance measure Fitts' law [53, 34, 8]. Mouse pointing performance in games has also been examined [4, 13, 14], with Claypool finding mouse target interception performance to be closely linked to interaction speed [14].

Gamepad

Game consoles require a specific gamepad, thus giving rise to many different versions of gamepads. Gamepad usage for PC games has been steadily increasing, with Steam reporting that more than 30 million players have plugged in a USB gamepad since 2015, with the vast majority of gamepads being Xbox or Playstation controllers [44].



Figure 2.5: Xbox 360 wired USB gamepad.

The most popular gamepads feature one or two thumbsticks for pointing, providing elastic indirect input and relative movement. Similar to joysticks, they typically sense force and control the velocity of an on-screen object (i.e. targeting reticle or cursor). Due to being elastic, thumbsticks

have a limited range of motion, thus making the maximum cursor speed game/application dependent on the assigned control/display ratio. Studies involving gamepads are relatively rare, although Claypool examined thumbstick target interception with latency and found latency to exponentially affect performance, especially above 250ms [11].

Touchscreen

Touchscreens have become ubiquitous in tablets and mobile phones, and are increasingly popular as monitors for desktops and laptops. Touchscreens use direct input and absolute movement, making them extremely intuitive. For example tapping the corner of the screen always results in a corner tap.



Figure 2.6: Dell 27” LED touchscreen monitor.

In terms of latency, touch latency was found to be important in user experience [1]. Commercial touch-screens can have up to 125ms of latency [42], which is considerably higher than the 2ms delay users are able to perceive as when dragging with direct touch input [58]. Yun et al. found touch latency can be significantly reduced if certain screen-related synchronization restrictions are relaxed [79]. These restrictions ensured that a display could never display information from multiple frames simultaneously (screen tearing), and that frame rates remained consistent (frame rate control). As for performance, previous research reports varying latency thresholds for when user performance of direct-touch dragging tasks begins to degrade (ranging from 25ms to 100ms) [42, 1, 79, 30].

Drawing Tablet

Drawing or graphics tablets provide a pen-like stylus used to touch the surface. The stylus allows for precise movements when being dragged across the tablet’s surface (with the user’s wrist often

resting on the same surface the tablet is resting on).



Figure 2.7: Wacom Intuos CLS 480s drawing tablet.

Drawing tablets feature indirect input, as the tablet and display are completely separate. However, the movement style can be changed to be either absolute or relative. In absolute movement mode, drawing tablets teleport the cursor to where the user taps by mapping corners of the surface to corners of the display. In relative movement mode, if you lift the stylus off the tablet and put it down in a different position, the on-screen cursor does not move (the same way as lifting the mouse up and setting it down does not move the cursor). Many tablets also feature a hover mode, where the stylus can control the cursor by hovering close to (but not touching) the tablet's surface.

Although less commonly used for gaming, tablets are of interest because some expert players of rhythm games (such as *Osu!*) eschew the mouse in favour of tablets. These experts primarily use the tablet in absolute movement mode. Drawing tablets allows for tapping without occluding the display space with your hand or stylus (you are not looking at the tablet surface when tapping). Additionally, absolute movement with its consistent mapping between input and output space may allow for easier memorization of tapping on sequences of targets. The adoption of drawing tablets by expert users shows that tablets can be used for competitive play.

In one of the few performance studies involving drawing tablets, Mackenzie compared dragging and pointing performance between mice, trackballs and tablets [52]. They found mice and tablets to have similar performance.

Cross-Device Comparisons

Since pointing devices all perform the same task, it is useful to compare and contrast multiple devices across a standardized task. Differences between device performance on a standardized task could yield valuable information, especially when extra latency is added. To our knowledge there is no single study that has compared a wide range of pointing devices under latency conditions.

Claypool combined results from a past study with a newer study to examine both mouse and

gamepad for interception tasks with latency conditions [11, 8]. Claypool then adapted Fitts’ law (a pointing performance measure) to model moving target interception with an exponential latency term. It is worth noting different participants were used in the various studies, and conditions varied (sometimes drastically) between studies.

As previously mentioned, Mackenzie et al. performed a cross-device study using mice, trackballs and tablets [52]. A later study by Mackenzie et al. expanded upon this and compared joystick, touchpad, trackball and mouse on a (Fitts’ law) pointing task. Mackenzie found the mouse to have the best pointing performance [51], and joystick to have the worst. Mackenzie also used this study to validate new pointing accuracy measures which we use later in chapter 3.

2.3 Games and Performance

If we are to measure latency’s effects on player performance, we must first define what *performance* is. There are many performance models for various tasks, though many are more generalized and not explicitly aimed at games. Below we list previously used laws and models for human performance on certain computing-related tasks.

2.3.1 Models of Human Performance

Fitts’ law is a famous law describing human motor movement. It has since become popular in a variety of fields including motor coordination and human-computer interaction. This law predicts the time it takes to point to a virtual on-screen target using a pointing device [23]. Each task is assigned an Index of Difficulty (ID) based on distance to and width of the target (we use the Shannon formulation: $ID = \log_2(D/W + 1)$ [49]). An Index of Performance (IP, or throughput) is calculated after the task has been attempted, and is based on both ID and task completion time (MT): $IP = ID/MT$ (in bits/s). A higher IP means the task was completed more quickly, in relation to the task’s difficulty.

This law has been modified heavily over the years, with various authors contributing modifications to suit different tasks. Jagacinsky et al. proposed an extension to model moving targets and additional terms for latency [40]. A linear latency term was proposed by Hoffman [34], whilst Mackenzie and Ware propose a multiplicative latency term [53, 77]. Claypool goes further, and suggests it should be exponential (similar to the formalism we later propose) [8, 11]).

Based on Fitts’ law, Meyer et al. further proposed that an aimed movement consists of a se-

quence of submovements towards a target [56]. An initial ‘open-loop’ primary submovement is first conducted, followed by increasingly smaller corrective ‘closed-loop’ secondary submovements. The initial movement is called ‘open-loop’ as there is no feedback available yet. Subsequent submovements are called ‘closed-loop’ as the previous movement is used as feedback for the next movement. Each submovement brings the movement closer to the target, thus closing the distance and making each movement smaller than the last.

When aiming under latency conditions, all visual feedback is delayed, either delaying the start of the next secondary submovement, or beginning the next submovement with incomplete information. This should negatively affect movement time of all ‘closed-loop’ secondary movements. However, the initial ‘open-loop’ primary submovement should not be affected by latency, since no (delayed) feedback is available yet.

Another law similar to Fitts’ law is the Hick-Hyman law, which characterizes the amount of time it takes a user to make a decision based on the number of possible choices they have [37]. Created and named after William Hick and Ray Hyman, they found decision time to increase logarithmically with number of choices. This finding only applies if the choices are sorted in some way, and does not apply if the choices are unsorted (thus requiring linear time to scan each one). The Hick-Hyman law can be applied to navigating sorted menus, and other software applications. As noted by Seow et al., both Fitts’ and Hick-Hyman laws have their roots in information theory, and share many similarities, however Fitts’ law has gained far more attention [73].

Another human performance model proposed by Card et al. is the Keystroke-Level Model (KLM), which as the name implies focuses on keystrokes on a keyboard. It estimates task execution time of an expert by breaking the task down into individual operation times [6]. The model is composed of six operators, four physical, one mental, and one system response; keystrokes, pointing, homing/moving the hands, drawing, and mentally preparing for future actions. Various computer tasks can be modelled using these operators. Variations have been proposed, such as extending the model to work on mobile phones [46] and in vehicles [64]

GOMS (goals, operators, methods, selection rules) [41] is yet another popular human performance model. Initially developed for editing text, it has been successfully applied to many HCI tasks. One example task of applying the model would be to edit a paragraph of text written by a colleague. The overall goal is to edit a colleague’s paragraph, but the goal could be split into sub-goals: edit the text, then send text back to colleague. Operators (the O from GOMS) are actions used to accomplish a goal. Operators may be perceptual, cognitive, or motor, and change either

an internal mental state or external physical state. The execution time of each operator can be represented using distributions, or a function of some sort (pointing can be represented with Fitts' law). Methods are potential sequences of operators meant to accomplish a single goal. One method for deleting a paragraph is clicking and dragging the mouse over the paragraph, and hitting the delete key. There are often many methods for accomplishing a goal, however only one will actually be chosen and used. Selection rules specify how a method is chosen. Users will develop personal preferences or heuristics, which then turn into selection rules. Continuing the paragraph deletion example, an example selection rule would be if the paragraph is too long (more than six lines), use the following method instead: double-click to select the text and hit the backspace key.

Sometimes more specialized models or adaptations are proposed for more specific tasks. Cockburn et al. proposed SDP (search, decision and pointing) as one such model, which specifically predicts pointing performance in menu selection tasks [17]. It captures the transition of user performance from novice to expert. Novices primarily rely on visual search, whereas experts graduate to using spatial memory (as governed by the Hick-Hyman law [37]).

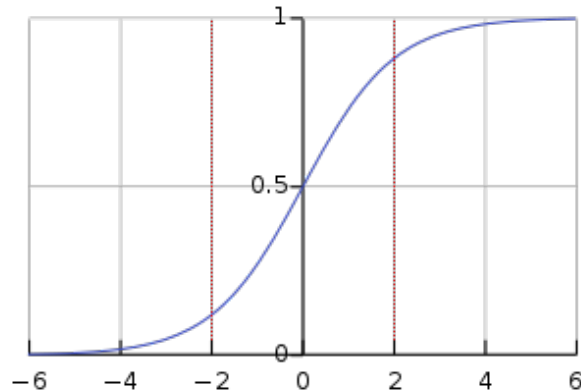


Figure 2.8: Example s-shaped sigmoid curve with X centered at the inflection point.

Lastly, psychometric functions describe the relationship between physical stimulus and forced-choice responses. Just noticeable differences (JND) are an example of this, where users report whether or not they noticed a difference in a stimulus such as hearing a sound, feeling a shock or difference in temperature [55]. The JND threshold is defined as being noticeable half the time. Psychometric tasks generally have a performance floor, where nothing is detected, a transition zone with decreasing error rates (or critical threshold), and a performance ceiling, where the stimulus is always detected. Modelled using generalized linear model, psychometric functions produce an S-shaped sigmoid curve (Figure 2.8). These functions are used extensively in chapter 5.

2.3.2 Game Atoms of Interaction

Similar to the GOMS model listed above, games can be broken down into small tasks, or ‘atoms of gameplay’. These tasks are small and repeatable, and are the main interactions inside a game. Example atoms include clicking on a target in a shooter, or blocking an attack in a fighting game. Different atoms require different actions to successfully complete, and may be more or less sensitive to latency.

Since games make constant repeated use of these atoms (especially ‘core’ interactions), it is important to understand how latency affects them. If a task is severely impacted by low levels of latency, or even expected levels of latency, the task may need to be altered. It is also useful to know how a task will be affected when under high amounts of latency as a worst case. Much of the delay comprising local latency is outside the control of the developer, however the game must still be fun to play and maintain a certain level of player performance even in worst case latency conditions. Below we review previous work related to gameplay and game tasks.

Very broadly, videogames are categorized into different genres. Game genres are often reported and decided by the developer, and there are no hard and fast rules as to what constitutes being in one genre over another. Generally, a genre will indicate what type of gameplay is to be expected (e.g. fast-paced shooting in a first-person-shooter), although gameplay can vary greatly between games of the same genre. Yuan et al. identify and use eight seminal game genres in their game accessibility survey: FPS, strategy, sports, RPG, puzzle games, racing games, dance/rhythm games, and adventure games [78].

Claypool uses the idea of determining a game’s sensitivity to latency using its genre by creating two measures: precision and deadline [12]. Deadline is the amount of time given for task completion, and precision is the level of accuracy required for the task. Actions with shorter deadlines and higher precision are more sensitive to latency. Using a sniper rifle in a shooter requires high precision to hit your target and has a short deadline as targets often move quickly across your sight, whereas constructing a building in a strategy game has looser deadlines since it takes a substantial time to complete, and building placement normally need not be precise. Claypool found game camera perspectives such as first-person, third-person, or from above to group and categorize most actions. Claypool found first-person games required higher precision and tighter deadlines, third person games required lower precision and moderate deadlines, and omnipresent games with the camera positioned above (such as in strategy games) requiring the least precision and loosest deadlines.

Claypool made these observations from examining Warcraft 3, Madden 2004 and Unreal Tournament 2003 [10, 59, 4].

These studies are useful for categorizing game genres and placing them on a general scale of deadline and precision. However, this scale does not tell us the exact effects latency will have on each game genre, only that certain genres are likely to be more sensitive to latency than others. Additionally, there can be much gameplay variation between games of the same genre. This leaves room for examination of game atoms themselves, and the developing of latency-performance predictive models.

2.3.3 Subjective Player Experience

Games are designed to create experiences. Testing a player’s skill and performance is only one part of this experience. All game elements such as art, sound, story, gameplay, and level design can affect a player’s experience, however since experience is subjective, different players find certain elements more important in terms of player experience than others. Positive player experience can cause a state of flow, fully immersing the player in the game. A common attribute of flow is the loss of one’s sense of space and time (“oops, I’ve been playing Civilization for 6 hours!”¹).

Player experience is how the player felt while playing, and while subjective, scales can be applied to retrieve measurements of player experience. Each scale must be validated to ensure each question is both useful and unbiased before it can be used in other work. Since latency has been shown to negatively affect not only performance, but also player experience, our studies make use of a number of validated scales. Below we list popular player experience scales, with some measures being used later in our studies.

The Player Experience of Need Satisfaction (PENS) scale is used to predict not only fun/enjoyment, but also game ratings, sales, developer loyalty, and sustained player interest [69]. The authors cite three major intrinsic psychological needs for generating motivational energy: competence, autonomy, and relatedness. Games must satisfy the need for competence through increasingly difficult gameplay. The need for autonomy explains why players enjoy freedom/choices in games. Lastly, relatedness is how players form relations between each other (though not all games are multiplayer). Through multiple studies on various game genres, six subscales were decided upon, which can be combined or used independently: Competence, Autonomy, Relatedness, Presence, and Intuitive Controls.

¹From personal experience of author.

The Intrinsic Motivation Inventory (IMI) focuses on user experience, and is a device used to measure experience whilst performing a target activity in laboratory experiments [68]. Based on self determination theory [72], which divides motivations into intrinsic (self) and extrinsic (external), the IMI ignores extrinsic motivations, as the user has already agreed to perform the task. Different versions of the scales have been proposed, but we use the following four sub-scales: Interest-Enjoyment, Perceived-Competence, Effort-Importance, and Tension-Pressure.

Attribution theory assigns causes to events, and how emotions and motivations are affected by these attributions [43]. This is important for games, as players may attribute their success or failures internally or externally. Ideally, game outcomes should be based on player performance, rather than some external factor such as luck (or latency). This is related to a sense of ‘fairness’, as unfair wins or losses are not desired. Depping et al. present a validated attribution theory scale applied to games [19]. They created four sub-scales: internality, stability, globality and controllability.

The NASA task load index (TLX) was developed to measure subjective workload across various tasks [28]. The NASA TLX enjoys widespread usage, with one survey study summarizing its usage in 550 other studies [27]. Though modifications exist, generally seven subscales are measured on a scale from low to high (except for performance, which is good to poor): mental demand, physical demand, temporal demand, performance, effort, frustration.

CHAPTER 3

EFFECTS OF LATENCY ON POINTING DEVICES (STUDY 1)

Games can be deployed on a wide variety of platforms, including PS4, Xbox One, Switch, PC, 3DS, smartphones, tablets or even Alexa (voice-controlled games). Some games are released on multiple platforms such as the popular battle-royale shooter Fortnite, which is available on PC, PS4, Xbox One, Switch, Android, and iOS. Additionally, a growing subset of multi-platform games offer online multiplayer between versions of the same game on different platforms (e.g. one player on PC playing online with another player playing the same game on a PS4). Popular examples of online cross-multiplayer games include Fortnite, Rocket League, Minecraft and Ark: Survival Evolved.¹

With multi-platform and cross-multiplayer games on the rise, it is now more important than ever to examine how input devices are affected by latency. Each platform has its own subset of input devices for game designers to choose from, with some devices being exclusive to that one platform (e.g. Wii U’s unique tablet controller). Selecting an input device to use can be difficult, as typical settings of play can differ greatly between platforms, with differing amounts of local latency being present. It is also likely that there are interactions between input devices and gameplay tasks — an extreme example would be that a mouse is likely to have better pointing performing than an etch-a-sketch with two one-dimensional controllers. The latency resistance and performance aspects of both input devices and core game interactions are especially important for cross-platform games, where players may get unfair advantages by playing on certain platforms and with certain devices.

Of the many different types of available input devices, pointing devices are among the most prominent. Common pointing devices include mouse, touchscreen and gamepads (also known as controllers). As discussed in the related work, pointing devices can vary greatly depending on certain properties such as direct/indirect input, absolute/relative movement, and property

¹Portions of this chapter appeared in previous published work: Michael Long and Carl Gutwin. Effects of Local Latency on Game Pointing Devices and Game Pointing Tasks. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI’19.

sensed/controlled. These different properties may affect how much pointing performance is lost under latency (latency sensitivity). This makes it critical to understand how pointing-related game interactions such as targeting, and target interception are affected by latency.

Static target acquisition (targeting or pointing) is common in both games and non-game applications. Game examples of static target acquisition include clicking on objects in ‘clicker’ or Candy-Crush-style games, selecting towers in tower defence games and stationary unit selection in real-time strategy games.

Interception, or moving target acquisition, is most commonly found in games, with real-world examples including shooting ducks in Duck Hunt, hitting falling objects in Fruit Ninja, aiming in side-scrollers/top-down shooters, and selecting moving units in real-time strategy games.

As discussed in the related work, there have been few multi-device studies examining pointing under the effects of latency. To remedy this, we needed to perform our own study to provide game designers with more knowledge about pointing devices and their relationship to latency. This study needed to answer a number of questions:

1. What are the performance effects of different levels of latency on different devices and different tasks?
2. Are there interactions between device and task in terms of performance?
3. Can the details of cursor or pointer movement (cursor paths) explain some of the performance results?
4. Are the effects of latency, device and task on specific aspects of player experience?

We performed a study comparing four input devices. Input devices were chosen to represent common gaming devices whilst covering a broad range of input characteristics. Chosen devices included touchscreen, drawing tablet, mouse and gamepad. Tasks included static target acquisition and moving target interception. Player performance and select player experience measures were recorded, alongside cursor path metrics.

3.1 Methods

3.1.1 Input Devices Chosen For The Study

Due to time constraints, we decided to evaluate four different input devices. In addition to wanting to cover a broad range of device characteristics described in subsection 2.2.1, we also consider the importance of each device in terms of gaming as a whole. Below we list the four chosen devices, along with explanation for their inclusion.

- **Mouse:** The mouse is the default device for gaming on the PC, and is used as a performance benchmarks for many pointing tasks [53, 34, 8].
- **Touchscreen:** Mobile gaming from smartphones and tablets featuring touchscreens has exceeded revenue from both console and PC gaming combined [57], and it would be remiss not to include such an important pointing device. Additionally, 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.
- **Gamepad:** In addition to being required to use an Xbox One and Xbox 360, the Xbox gamepad is also increasingly popular for use on PC. Although the thumbsticks provide similar indirect input and relative movement as the mouse, the thumbsticks are elastic and rate-controlled, providing a significant difference in experience.
- **Drawing Tablet:** The drawing tablet was included for its unique input characteristics and its use by top players of popular rhythm games such as Osu!. For additional relevance, both PS4 and Steam controllers have touchpads which can function similar to a drawing tablet.

3.1.2 Cursor Path Metrics

Mackenzie identified measures related to the path of travel during a pointing task [51]. These measures provide a richer description of pointing tasks than simply analyzing Fitts' Index of Performance (IP) 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 3.2). When describing these path metrics, y is a sample point of the actual cursor path. These measures and




	Direct Input	Indirect Input
Absolute Movement		
Relative Movement		

Figure 3.1: Input devices used, divided by device properties.

their implementation are reported below for ease of reference as they are an integral part of this study.

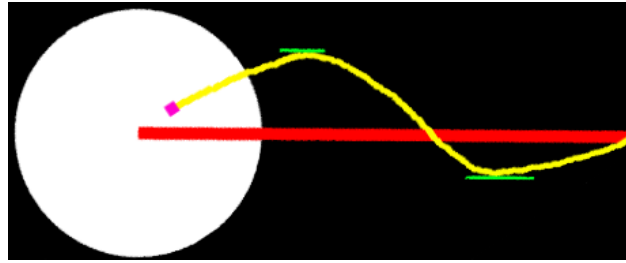


Figure 3.2: Cursor (pink) and its path (yellow) along task axis (red) towards target (white circle). This path features one task-axis crossing, and two movement direction changes (green).

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 the 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 orthogonally relative 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 mistaken clicks made with cursor outside of target.

3.1.3 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, with only two participants regularly using 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. Participants had an average response time of 276ms to a single simple stimulus [36], and a median response time of 267ms (visualized in Figure 3.3).

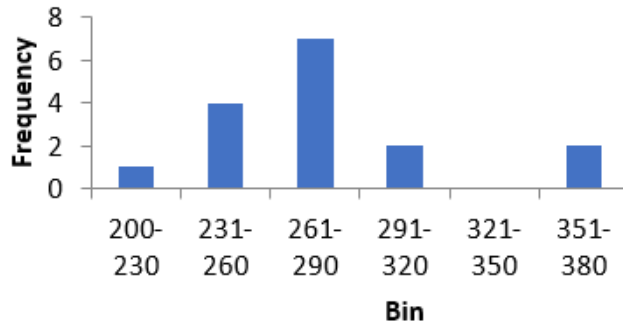


Figure 3.3: Response time histogram of participants in milliseconds (ms).

3.1.4 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 133Hz, 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.

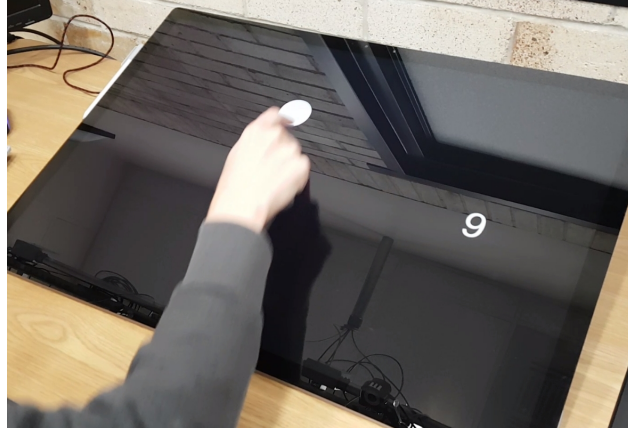


Figure 3.4: Experimental setup using touchscreen.

Following procedures from prior work [39], 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, with a standard deviation of 6ms (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 3.2) was simulated at the input device level by buffering input for later use.

3.1.5 Procedure

Participants first completed informed consent and demographic questionnaires, and performed a response time test [36]. 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 (Tables 3.1 and 3.2). 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.

3.1.6 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

Question	Source
Q1: I felt capable and effective when playing.	PENS: Competence [69]
Q2: Using the current input device was fun.	IMI: Enjoyment [68]
Q3: I put a lot of effort into those rounds.	IMI: Effort [68]
Q4: How well I did was completely due to me.	Attribution [19]
Q5: I was frustrated by the task.	TLX: Frustration [28]
Q6: The cursor movement was responsive.	Custom

Table 3.1: In-game experience questions asked after every second latency level.

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

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 3.2), and an input device (mouse, controller, touchscreen or drawing tablet). Device order was balanced using a Latin square.

Target position, size, 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 3.5). Data was not recorded from practice rounds.

3.2 Results: Stationary Targets

3.2.1 Movement Time

As seen in Figure 3.5, 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 3.6 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 decreased at the fastest rate. The right side of Figure 3.6 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 $-.25$, $p < .001$).

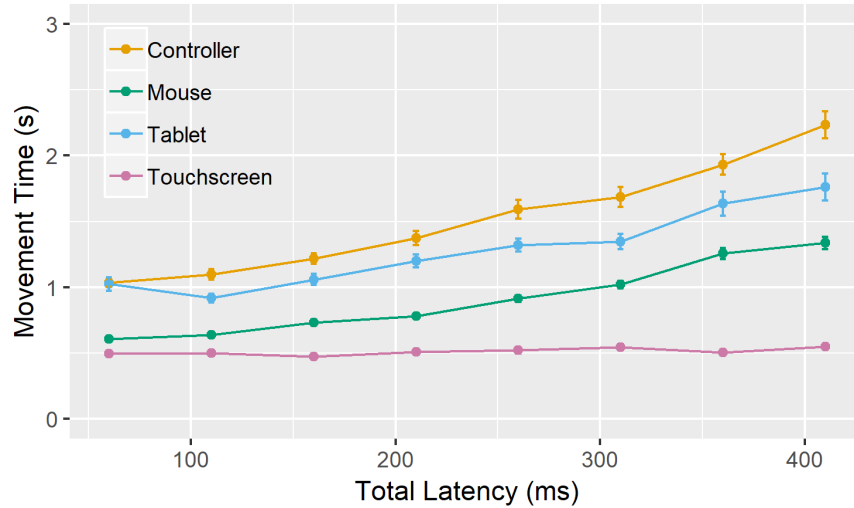


Figure 3.5: Movement Time (MT) for stationary rounds. (MT = Duration - Latency).

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 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 $p < .01$).

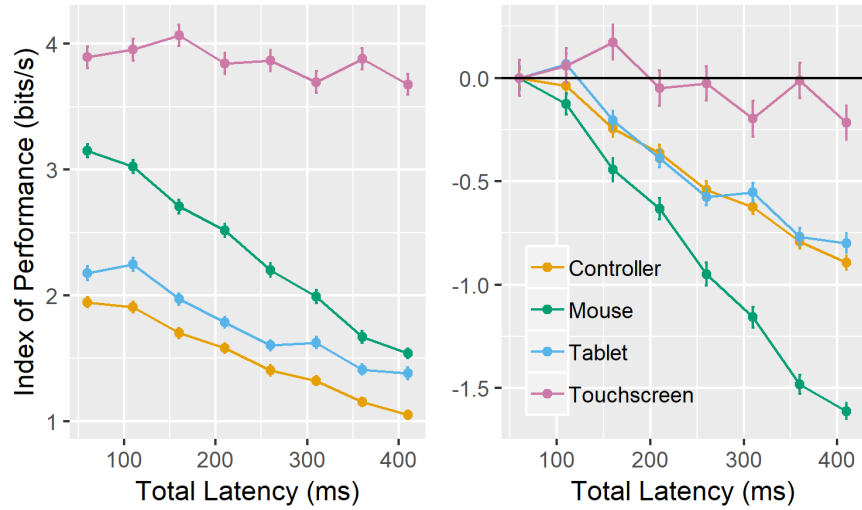


Figure 3.6: Left: IP of stationary rounds. Right: IP of each device relative to performance at 60ms total latency (black line).

3.2.2 Path Metrics

Cursor paths are visualized in Figure 3.7, and are useful for interpreting numeric path metrics in Figure 3.8. 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 controller’s thumbstick produced paths with straight sections that rarely changed movement direction that heavily diverged from the optimal path (especially at higher latencies). At low latencies, the controller had few misclicks and target re-entries (TRE), however

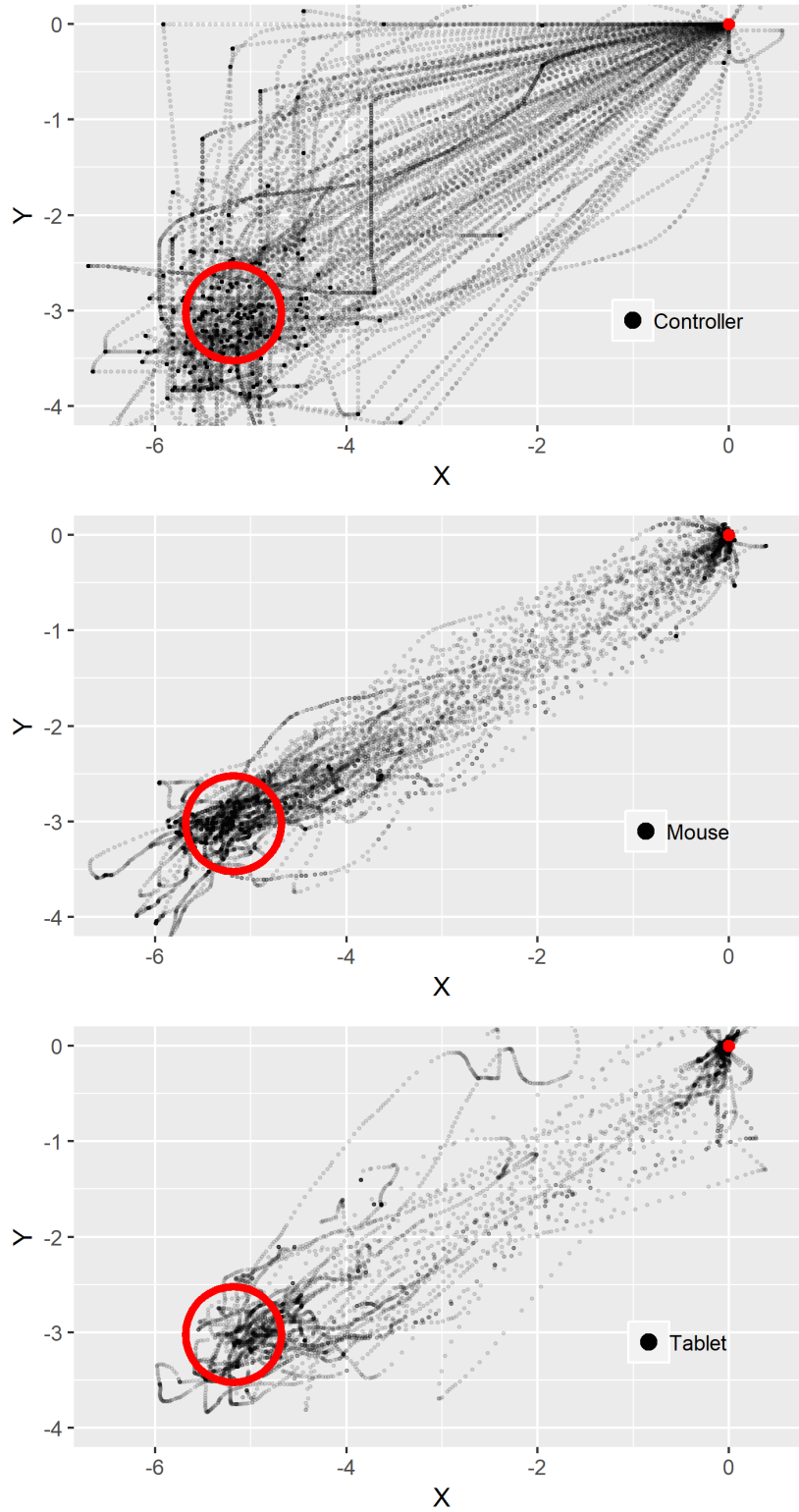


Figure 3.7: Random selection of 215 cursor paths of stationary rounds with varying latencies (target width 1, target distance 6, ID of 2.81). Cursor starts at red dot in top right, and moves towards red target circle in bottom left.

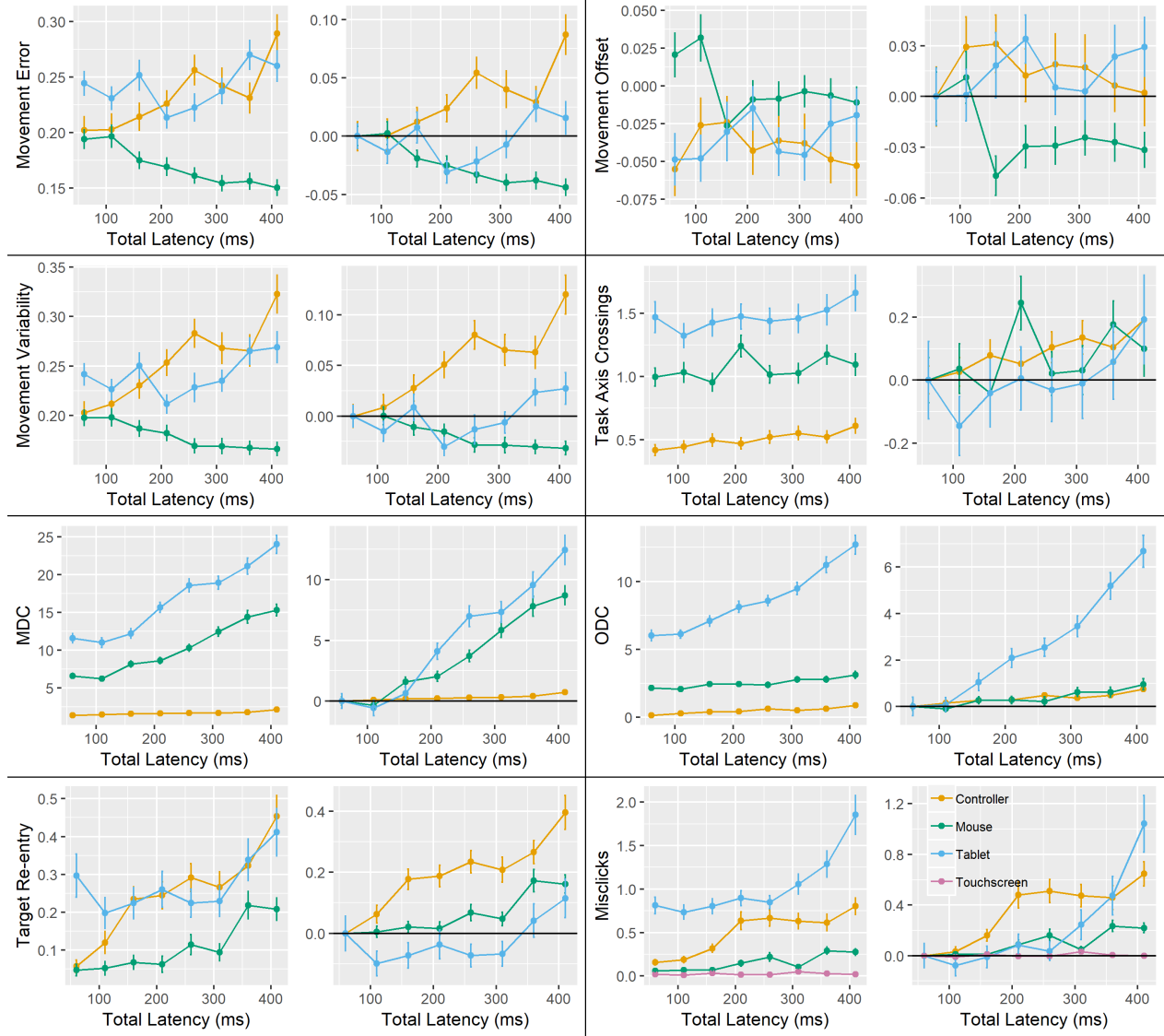


Figure 3.8: Left of each pair of charts: path metrics for stationary pointing tasks. Right of each pair: path metrics relative to performance at 60ms. From left to right, top to bottom: movement error (ME), movement offset (MO), movement variability (MV), task axis crossings (TAC), movement direction 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.

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 decrease 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 3.8). 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 (Table 3.3) 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

	IP	ME	MO	MV	TAC	TRE	MDC	ODC	Misclicks
IP	–	-0.47	0.08	-0.49	-0.28	-0.65	-0.36	-0.33	-0.31
ME		–	-0.15	0.94	0.15	0.52	0.19	0.34	0.31
MO			–	-0.09	0.01*	-0.08	-0.01*	-0.01*	-0.08
MV				–	0.22	0.55	0.17	0.33	0.30
TAC					–	0.44	0.51	0.45	0.30
TRE						–	0.48	0.47	0.49
MDC							–	0.64	0.38
ODC								–	0.47
Misclicks									–

Table 3.3: Pearson’s R cross-correlations between path metrics and IP. All $p < .05$ except for entries denoted by *. Top row, left to right: Index of Performance (IP), Movement Error (ME), Movement Offset (MO), Movement Variability (MV), Task Axis Crossings (TAC), Target Re-entry (TRE), Movement Direction Change (MDC), Orthogonal Direction Change (ODC), Misclicks.

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 with IP across all devices and latencies (R of -0.66), 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 and cursor movements when the pen was brought down for a tap.

3.3 Results: Moving Targets

In moving-target tasks, the target had a speed of 1200 pixels/sec (158mm/s). 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

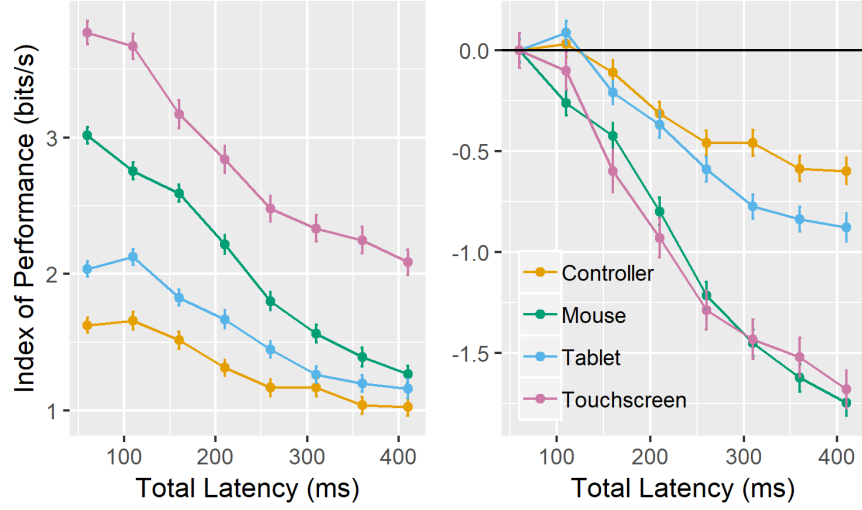


Figure 3.9: Left: IP of moving rounds. Right: IP relative to performance at 60ms (black line).

speeds [14]. Latency was found to be significantly and negatively correlated with IP (Pearson’s R of $-.36$, $p < .001$). Repeated-measures 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 3.9). All three indirect devices (mouse, tablet, controller) had roughly the same IP at 410ms latency.

When examining relative performance losses (right side of Figure 3.9), IP for both the mouse and touchscreen degraded far more steeply than the controller and drawing tablet (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) and misclicks (MC). Both TRE and MC (Figures 3.10 and 3.11) rapidly

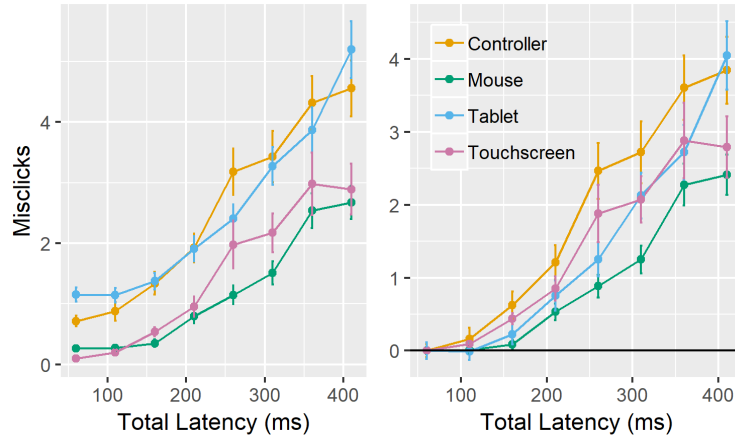


Figure 3.10: Misclicks of moving rounds. Touchscreen included.

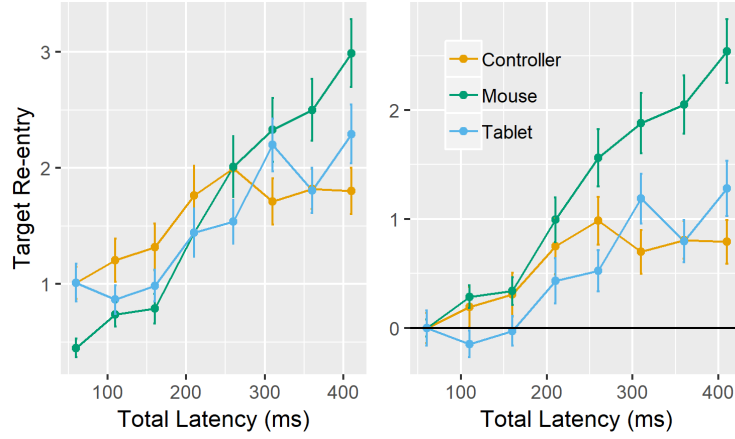


Figure 3.11: Target re-entries (TRE) of moving rounds.

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).

3.4 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 3.12). 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 3.6 and 3.9).

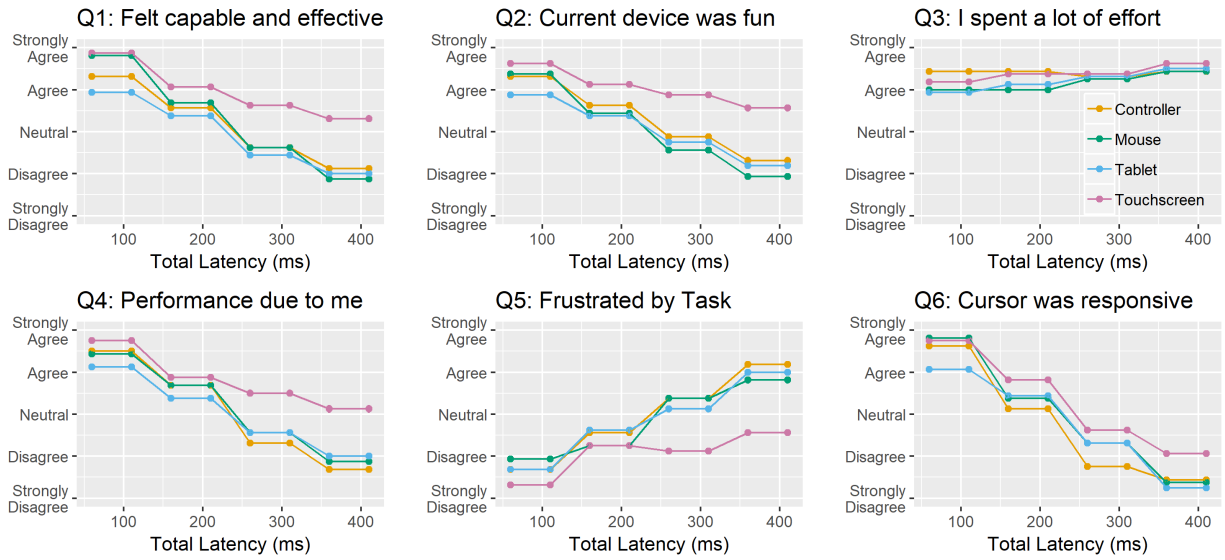


Figure 3.12: Survey questions on a 5-point Likert scale with standard error bars. Higher means agreement with the question. Survey asked after every second level of latency.

After the study was concluded, participants were asked which device they preferred, and which was most resilient to lag, with 13 out of 16 choosing the touchscreen for both questions (Table 3.4). When asked about strategies, most players mentioned predicting or leading target movement. Several mentioned they clicked more rapidly under high latencies, often clicking before the cursor had reached the target (especially in moving target rounds). Some participants adopted a two-handed approach when using the touchscreen, with each hand covering half the screen.

3.5 Summary

We now recap and summarize overall results for each device. Design guidelines based on these results are presented in chapter 7.

Device	Preferred Device	Most Resilient Device
Touchscreen	13	10
Mouse	2	3
Drawing Tablet	1	2
Controller	0	0

Table 3.4: Participants were asked which device they preferred using, and which was most resilient to lag.

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. Most participants would be considered expert mouse users, making it unsurprising that users could perform

quick and precise movements using relatively little physical space. This may have led to users overcompensating less when lag was present.

Touchscreen

The touchscreen had the best performance, and was unaffected by latency for stationary pointing. The touchscreen also performed well with interception tasks, however performance rapidly decreased as latency increased. Strong overall performance and freedom from latency effects for stationary targets makes the touchscreen the best examined device for games that involve pointing whilst dealing with local latency.

Path Metrics

By analyzing device-specific cursor path metrics, we determined the most important path metrics (in relation to throughput) and a number of desirable pointing device traits. We found target re-entry (TRE) to be the single most important path metric. To reduce target re-entry, ideal pointing devices must not be prone to slippage or overshooting. Cursor speed must be also 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.

CHAPTER 4

EFFECTS OF LOCAL LATENCY ON GAME SPEED AND GAME ATOMS (STUDY 2)

Game speed determines the deadline by which a task must be completed. Fast game speeds require fast reflexes and split-second decision making. Shooters, fighting games, rhythm games and endless runners such as Geometry dash are all examples of game genres with fast game speeds. Game genres with typically slower game speeds include puzzle, RPG, turn-based strategy and adventure games. Even games within the same genre can vary greatly on game speed — a shooter with one-shot kills such Counterstrike requires faster reflexes than Borderlands, where it takes multiple shots to defeat an enemy. Based on game speed and the precision required to complete a task, Claypool identifies a number of genres that are likely be more sensitive to latency [13]. This is useful as a general guideline, but we still do not know the exact effects that latency and game speed will have on gameplay.¹

Interactions within a game can be further broken down into smaller and more easily measured “game atoms”. Game atoms include tasks such as pointing, tracking, reaction, or selection. Since games make constant repeated use of these atoms, it is important to understand how latency affects player performance in an atom. Player performance can suffer under even low amounts of latency if the given game atom is particularly sensitive to latency.

In order to give game designers information regarding latency and game speed’s effects on game atoms, we performed a study that answered the following questions:

1. Hoes does local latency affect player performance of a game atom?
2. How does game speed affect player performance of a game atom?
3. Are there any interactions between latency and game speed?

¹Portions of this chapter appeared in previous published work: Michael Long and Carl Gutwin. 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)* '18, pages 285–297, New York, NY, USA, 2018. ACM

4. How does latency and game speed affect specific player experience measures?

We performed a study examining how player performance in a small and simple game atom is affected by both latency and game speed. We examined a small, simple and common game atom — 1D target interception with a mouse. The study is based on Pong, a simple and iconic game featuring 1D target interception.

In this chapter, we first describe what a game atom is, and how to identify one. Then we report on our study, and analyze its results. Last, we present a new measure called Time to React that combines both game speed and latency.

4.1 Identifying Game Atoms

Generally, designers will have intuitions as to what constitutes a game atom in their game. For the purposes of predicting error rates using our model, we are interested in short and simple atoms that would be affected by latency. Below we list traits that a game atom ideally has:

- Small or short
- Simple
- Repeatable
- Has a measurable outcome (e.g. succeeded or failed, time to completion)

The measurable outcome of the atom can often be altered depending on your definition of success. For example, clicking on a button could have its outcome be measured as the time it takes to click the button, or it could be rephrased with a time limit; the user has 5 seconds to click on the button. This changes the outcome from a continuous value to binary win/loss.

Game atoms can be thought of as either a lower-level action, or as game-specific implementations of these higher-level actions. Lower-level actions include pointing, target interception, key presses, menu navigation, reaction, selection or some combination thereof.

Game-specific implementations of the above actions often include more variation, and can be more complex — specific timings, other specific game knowledge (animation or sound cues) and interaction effects from various goals (shoot enemies whilst conserving ammo and avoiding damage) can make these implementations harder to generalize, thus requiring more analysis. Examples of game-specific implementations include aiming in an FPS, blocking an attack in a fighting game,

jumping over a pit in a side-scroller, throwing and arcing a grenade in a FPS, aiming in a side-scroller, and equipping an item in an RPG. Claypool lists a number of higher-level player actions from various game genres which could likely be broken down into multiple game atoms depending on their complexity and implementation: racing, running, combat, drinking potions, fighting, moving, building, and exploring [12].

4.2 Methods

4.2.1 Game Design

We built a custom ball-return game based on the arcade classic Pong (Figure 4.1). 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.

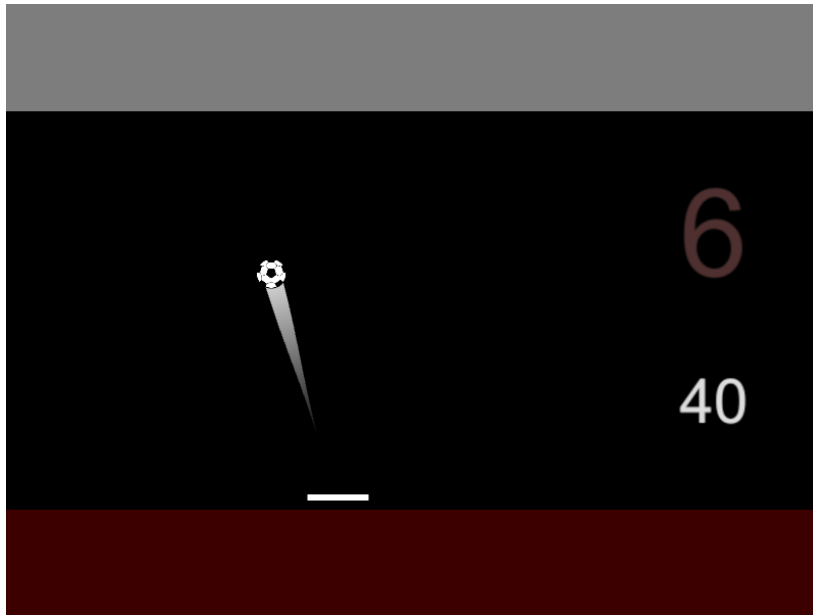


Figure 4.1: Study game. Player controls the white paddle; the player has just returned the ball upwards. White number indicates time remaining; red number shows return streak.

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.

4.2.2 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 videocard, an MSI Interceptor DS100 mouse (1000Hz), and a 24” (1920x1080) BenQ 120Hz HD monitor. Mouse sensitivity was set to 800dpi, and Windows mouse acceleration was disabled. The game ran at a constant 120fps, matching the refresh rate of the monitor.

Following procedures from prior work [39], baseline 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 with a standard deviation of 6ms (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.

4.2.3 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 in a typical week (mean 6.6 hrs/week), and seven reported having experienced latency in games. Participants had an average response time of 388ms to a single simple stimulus [36], and a median response time of 333ms (visualized in Figure 4.2).

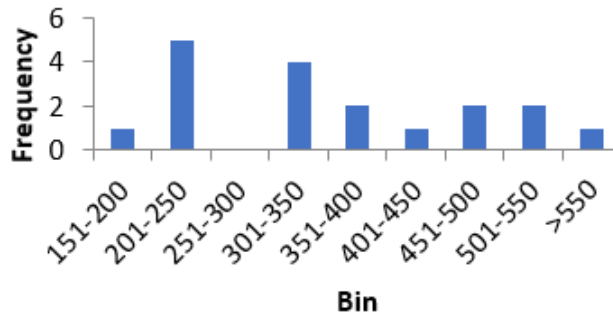


Figure 4.2: Response time histogram of participants in milliseconds (ms).

Round	1 P	2	3	4	5	6	7
Added Latency (ms)	0	50	250	0	200	100	150
Total Local Latency (ms)	49	99	299	49	249	149	199

Table 4.1: Local latencies for each round; P=practice.

4.2.4 Procedure and Study Conditions

Participants completed informed consent and demographics questionnaires, and performed a response time test [36]. 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.

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 4.1 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 latency on play experience, using questions from several sources (Table 4.2).

Question	Source
Q1: I felt capable and effective in that round.	PENS: Competence [69]
Q2: Playing that round was fun.	IMI: Enjoyment [68]
Q3: I put a lot of effort into that round	IMI: Effort [68]
Q4: How well I did during that round was completely due to me.	Attribution [19]
Q5: During that round, the movement of the paddle was responsive.	Custom
Q6: I was frustrated during that round.	TLX: Frustration [28]

Table 4.2: Experience questions asked after each testing round.

4.2.5 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).

4.2.6 Performance Results

Error rates roughly ranged from 0.06 to 0.8 (see Figure 4.3) 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* ($F_{15,85} = 542.46$, $p < .001$, $\eta^2 = .83$) and *game speed* ($F_{2,34} = 367.04$, $p < .001$, $\eta^2 = .61$) on error rate. ANOVA also showed a significant interaction between *local latency* and *game speed* ($F_{10,170} = 39.03$, $p < .001$, $\eta^2 = .44$). (We report the effect size for significant RM-ANOVA results as general eta-squared, considering .01 small, .06 medium, and $> .14$ large [21]).

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.

4.2.7 Combining Speed and Latency into 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

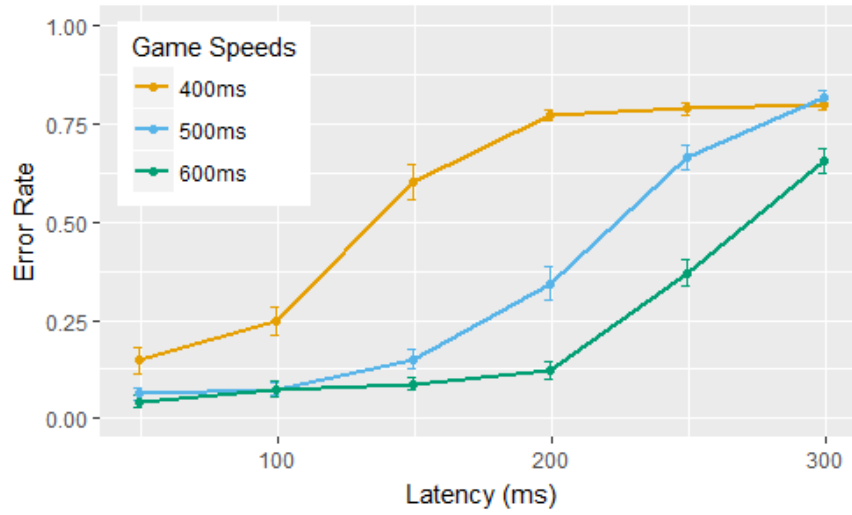


Figure 4.3: Error rate vs. local latency with standard error bars.

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 4.4 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 4.3).

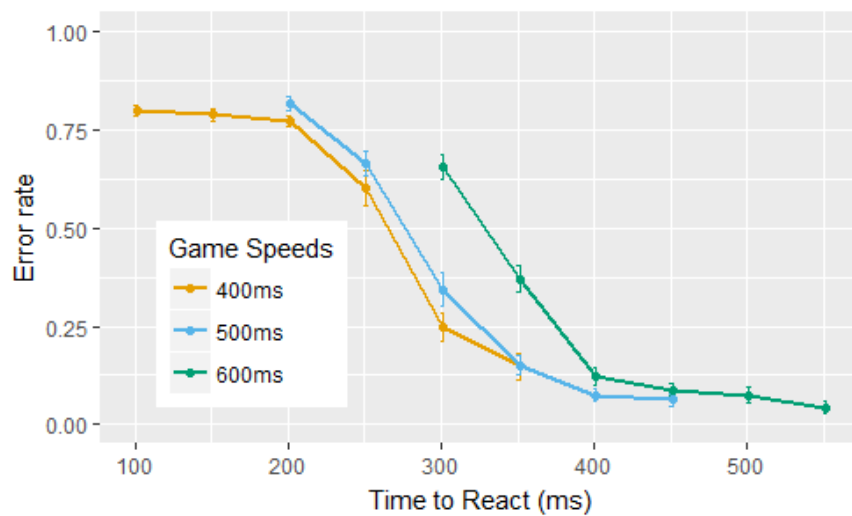


Figure 4.4: Error rates vs. TTR (game speed minus latency), by game speed. Note different curve directions compared to Figure 4.3.

Figure 4.4 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 (results seen in Figure 4.5).

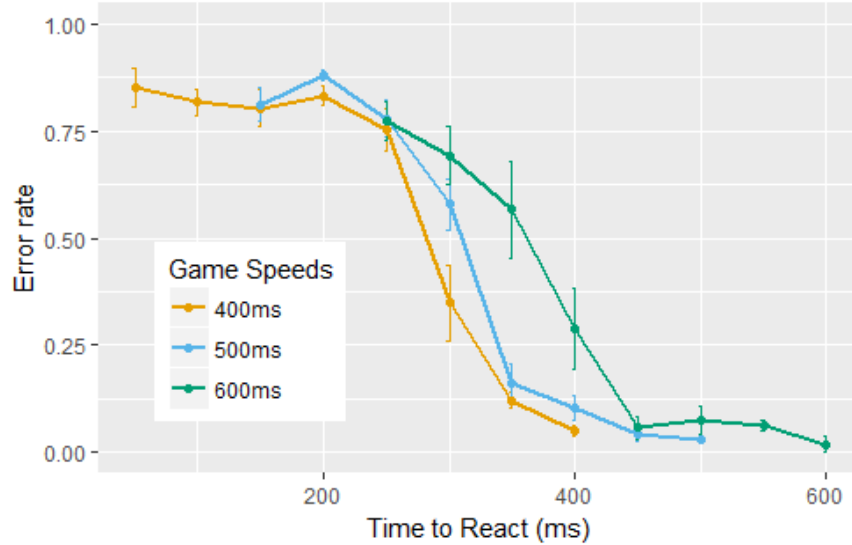


Figure 4.5: Error rates of small follow-up trial with added local latency levels 300ms and 350ms. Note the lines extend further to the left when compared to Figure 4.4.

4.2.8 Experience Questionnaire Results

Local latency had a substantial impact on almost all player experience questions (see Figure 4.6), except for Q3 (effort spent). Players felt less capable (Q1), had less fun (Q2), attributed success more externally (Q4), found the paddle less responsive as latency rose (Q5) and felt more frustrated (Q6). Local latency was significantly correlated with all survey questions ($p < .05$, 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 a similar sigmoid shape described above for performance data.

Participants felt less capable and effective as 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). Participants were not told if latency would be added in each round (or

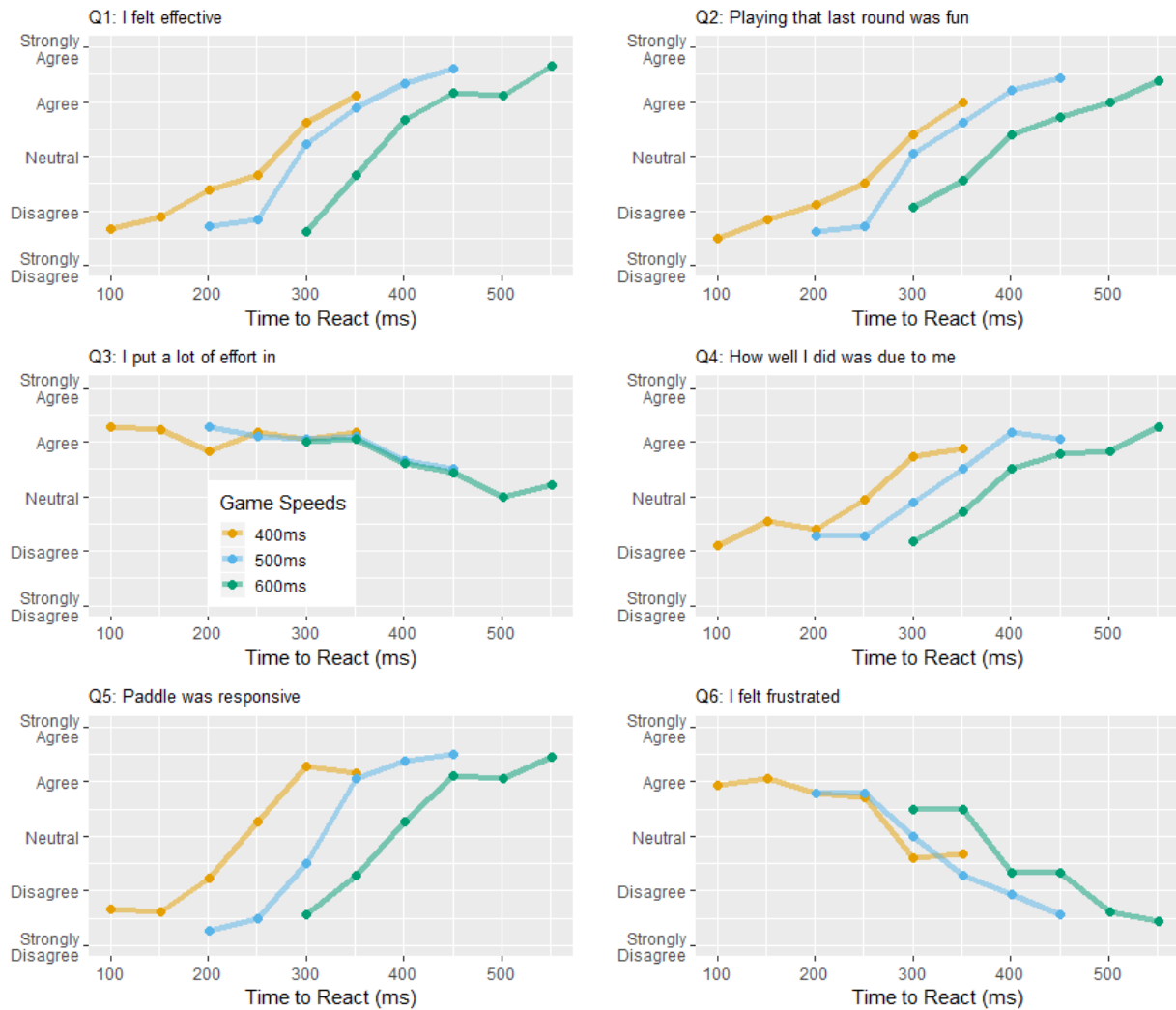


Figure 4.6: Responses to player experience questions (Table 4.2), vs. Time to React using 5-point Likert scales.

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.

As for the effects of game speed on player experience, Kruskal-Wallis tests showed significant differences between *game speed* on four of the six experience questions (Q1: $\chi^2 = 14.80$, Q2: $\chi^2 = 19.80$, Q3: $\chi^2 = 13.6$, Q4: $\chi^2 = 5.47$, Q5: $\chi^2 = 3.92$, Q6: $\chi^2 = 25.82$, all $p < .05$ except for Q4: $p = .07$ and Q5: $p = .14$). This demonstrates game speed to significantly affect many experience measures, including: capability and effectiveness (Q1), fun (Q2), effort invested (Q3) and frustration (Q6). Slower game speeds were found to have substantially more positive average player experience scores than faster game speeds (e.g. for Q2: Fun, values ranging from 1 (strongly disagree) to 5 (strongly agree), $600ms = 3.35$, $500ms = 3.11$, $400ms = 2.56$). It should be noted that game speeds were presented in order of slowest to fastest, and participants may have gotten tired of the task as the experiment continued.

CHAPTER 5

BUILDING AN ERROR-RATE PREDICTION MODEL

Game companies spend considerable time play-testing their game. Hiring professional testers and bringing in temporary testers is both expensive and time consuming. One of the most important aspects of play-testing is providing feedback on the difficulty of gameplay segments. Too high of a difficulty can yield negative player experiences by failing to provide players with a needed sense of competence [69]. An additional aspect that requires play-testing is the ‘responsiveness’ of a game — responsiveness means how quickly the game responds to input. Having a slow or unresponsive game will affect player performance, experience and game difficulty [7].¹

A non-subjective way of evaluating difficulty is through player performance represented by player error rates. Although some number of errors are expected to occur during a task, too high of an error rate can be frustrating. Both study 1 (chapter 3) and study 2 (chapter 4) used error rates as their primary performance measure.

Based on results from chapter 4, as well as previous work, we know that latency significantly and negatively affects error rates for many gaming and non-gaming tasks [66, 4, 39, 12, 20, 5, 75]. This demonstrates latency to be an important contributing factor to error rate, thus requiring extra play-testing at various latency levels to ensure good user experiences.

However, a task’s sensitivity to latency is also dependent on game speed. Both results from chapter 4, and Claypool’s work on target interception confirm game speed must be accounted for to determine latency’s effects [11, 8]. For example, faster tasks see a relative increase in errors due to latency over error rates of slower tasks.

To reduce latency-related play-testing, we built a predictive error rate model using data from chapter 4 (study 2). The model uses a logistic function to predict errors given task game speed, level of latency and other task-specific parameters. By providing an easy-to-use model, game designers

¹Portions of this chapter appeared in previous published work: Michael Long and Carl Gutwin. 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)* ’18, pages 285–297, New York, NY, USA, 2018. ACM

can easily test multiple game speeds to determine how sensitive a game atom is to latency. Play-testing resources can then be allocated more efficiently; problematic game atoms can receive more attention, whilst latency-resistant atoms receive less. Game speed of a task can also be adjusted, so as to reduce errors in latency-sensitive game atoms.

5.1 Creating a Predictive Model

The consistent sigmoid curves seen in the performance data of study 2 (Figure 5.1), suggest that the relationship between local latency and Time to React (TTR) could be modeled using a logistic function. Because our results showed 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 5.1 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.

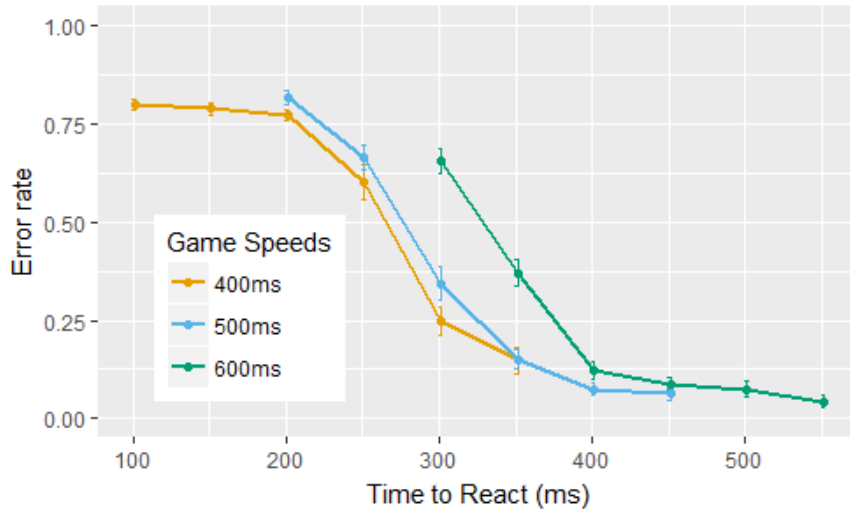


Figure 5.1: Error rates vs. Time to React of study 2. Note the S-shaped sigmoid curves. Reproduced from Figure 4.4 in chapter 4.

$$AdjustedTTR = Gamespeed - (Latency + \frac{Latency^2}{1000}) \quad (5.1)$$

We applied Equation 5.1 to our results and found that ATTR better grouped the curves together

than the standard TTR, especially at higher levels of local latency (Figure 5.2).

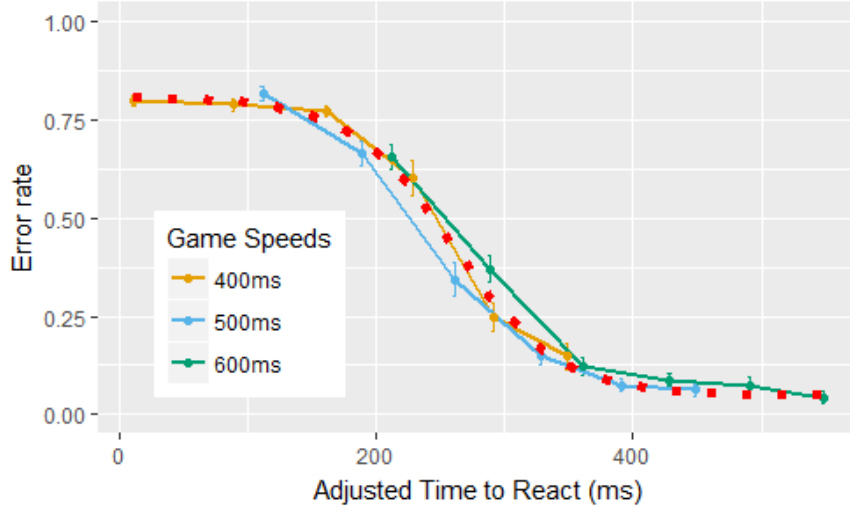


Figure 5.2: Error rates vs ATTR, with logistic model shown in red dots ($B = 0.05$, $L = 0.81$, $x_0 = 260$, $k = 0.025$, Equation 5.2).

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 5.2), 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 x_0 is the x value of the sigmoid’s midpoint.

$$f(x) = B + \frac{L - B}{1 + e^{-k(x_0 - x)}} \quad (5.2)$$

To fit a sigmoid curve, we choose values for the four variables: floor, ceiling, midpoint and 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 $0.05 * \#oftasks$. The ceiling should be set to the highest rate of errors that can be expected either if the player does nothing, or moves completely randomly. The ceiling may need to be adjusted if no randomness is involved in the task. For the game atom study (study 2, Pong, chapter 4), the ceiling (L) is a function of both paddle and ball width; the paddle took 6% of the screen width, and the ball 3%, however the ball could hit either the left or right side of the paddle, making the effective blocking width 12% of the screen. However, the random direction the ball moves is limited to be within 50% screen width of its position when it changes direction. This gives an 18% chance the ball is intercepted if the player positions the paddle within the correct half of the screen. We thus set the 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 a game atom. We approximate complexity as the number of tasks that the player must perform. The Pong game had a single task: move the mouse to block the incoming ball. For the midpoint (x_0), we estimate a base of 130ms plus an additional 130ms per task ($130 + 130 * \#oftasks$). 130ms was chosen as a base because it is roughly half the median reaction speed to a single simple stimulus [36], 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 $0.05 * 0.5^t$ 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.

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). Since our sigmoid curve model is non-linear, and R^2 assumes the model is linear, this score should not be taken too seriously, however we still provide for reference since there are few ways to evaluate goodness of fit for non-linear models. We also compared the error rates predicted by the model to the actual rates seen in the study (Table 5.1). At 300ms ATTR, our predicted error rate was within 1% of the actual error rates (Table 5.2). 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 current data well.

Game Speed	600ms	500ms	400ms
Mean Error Rate Difference	0.023	0.027	0.028

Table 5.1: Mean absolute differences (predicted - measured).

Adjusted Time to React	400ms	300ms	200ms
Measured Error Rate	0.096	0.273	0.679
Predicted Error Rate	0.077	0.268	0.682
Difference	0.019	0.005	-0.003

Table 5.2: Measured vs. predicted error rates by ATTR value.

5.2 Application of Predictive Model

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. 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. We list attributes of a game atom at the start of this chapter. 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 [39]). Note that latency does not need to be sensed during actual 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 play-tests.

Fourth, the designer calculates Adjusted TTR with game speed and expected local latency using Equation 5.1. Fifth, the designer needs to determine the four parameters for the sigmoid curve (Equation 5.2), with the floor and ceiling 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 5.2.

5.2.1 Steps to Apply Model

1. Choose the game atom to analyze
2. Determine game speed for this interaction
3. Determine local latency values for investigation
4. Calculate Adjusted Time to React using Equation 5.1
5. Choose parameters for sigmoid curve (Equation 5.2)
 - Ceiling (B): 0.95, or expected errors from chance
 - Floor (L): $\#oftasks * 0.05$
 - Sigmoid midpoint (x_0): $130 + 130 * \#oftasks$
 - Steepness of curve (k): $0.05 * 0.5^{\#oftasks}$

6. Calculate predicted error rate by plugging in ATTR as x into the parameterized version of Equation 5.2

CHAPTER 6

MODEL VALIDATION (STUDY 3)

In chapter 5 we created a model of errors based on local latency magnitude and game speed, but more validation is required before it can be deemed useful. Real-world games are generally more complex than Pong (chapter 4, study 2), and feature multiple simultaneous tasks that have interaction effects. For example, in a first-person shooter you are simultaneously moving to evade incoming projectiles whilst shooting at enemies. Performing each action in isolation grants better performance for that respective action (solely dodging projectiles versus solely shooting at targets), and performing both actions simultaneously generally means lower performance for both actions. Our model must take interaction effects into account. More complex games will have more simultaneous tasks, and more interaction effects.

To test the predictive model in a more complex game, we carried out a study based on a Space Invaders-like game. The game was more complex than the Pong game (chapter 4, study 2), 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 between 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/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.



Figure 6.1: Space Invaders game. The white ship is controlled by the player. The player must dodge the red bullets whilst shooting the moving green invaders. The red number on the right is the player’s score (enemies hit minus number of times player was hit), and the white number is time remaining in round.

Green invaders were arrayed along the top of the screen in two rows of 11. The player could fire by clicking or holding down 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 bullet hit an invader, and decreased when their ship was hit. Players were told to prioritize dodging enemy bullets over shooting invaders.

6.1 Methods

6.1.1 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 120fps, matching the refresh rate of the monitor.

Base local latency was measured in the same way as the Pong study (chapter 4) and was found to be 45ms, with a standard deviation of 3ms between recorded frames. All charts below involving latency and TTR have this 45ms of local latency included. Artificial latency was managed in the same way as the Pong study.

6.1.2 Participants

Twenty participants (9 female, 11 male, average age 25, median 23) were recruited from a local university, with all but one being students. All but one was right-handed, with 18 people using the mouse in their dominant hand. Sixteen people played videogames during a typical week (average of 7.5 hrs/week). All but one of the participants reported having experienced lag when playing videogames. Using an online reaction test to a single simple stimulus [36], participants had an average response time of 367ms to a single simple stimulus [36], and a median response time of 335ms (visualized in Figure 6.2).

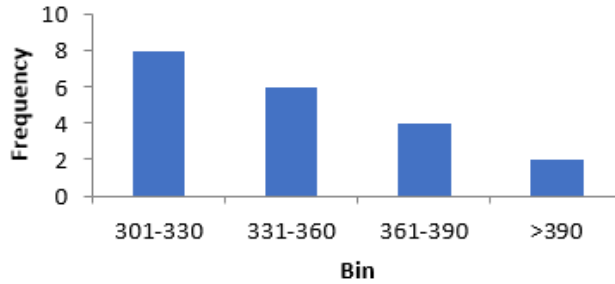


Figure 6.2: Response time histogram of participants in milliseconds (ms).

6.1.3 Procedure and Study Conditions

Participants completed informed consent and demographics questionnaires, and performed a response time test [36]. 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 6.1). 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 6.2).

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. In each group of six rounds, added latency started at zero and increasing in ascending order (Table 6.2).

Question	Source
Q1: I felt capable and effective when playing that last round.	PENS: Competence [69]
Q2: Playing that last round was fun.	IMI: Enjoyment [68]
Q3: How well I did during that last round was completely due to me.	Attribution Measure [19]

Table 6.1: Questions asked after each testing round.

Round	1 P	2	3	4	5	6
Added Latency (ms)	0	0	100	200	300	400
Total Local Latency (ms)	45	45	145	245	345	445

Table 6.2: Local latency in each round; P=practice.

6.1.4 Performance Measure

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 (in terms of percentage width of screen), multiplied by the spawn rate of enemy bullets per second (6.66/second).

6.2 Results

As shown in Figure 6.3, error rates follow sigmoid curves (although less so than study 2 in chapter 4), with performance roughly ranging from 0.1 to 1.0 errors/sec. The Space Invaders game was, however, more sensitive to local latency than study 2: 100ms of added local latency had an immediate and noticeable effect on errors for all speeds.

Error rate was significantly correlated with latency (Pearson’s R of 0.76, $p < .001$). Repeated-measures ANOVA again showed significant effect of *latency* ($F_{4,76} = 189.19$, $p < .001$, $\eta^2 = .66$) and *game speed* ($F_{3,57} = 36.23$, $p < .001$, $\eta^2 = .18$) on *error rate*, with a significant interaction

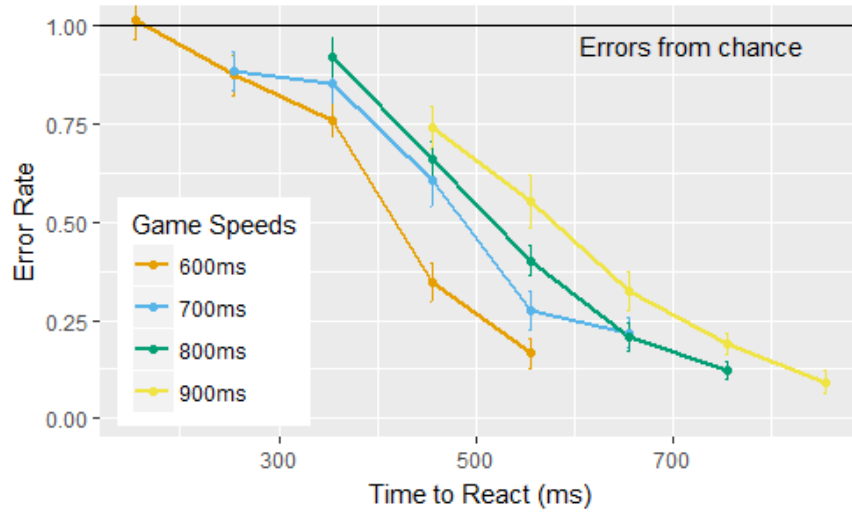


Figure 6.3: Error rate vs TTR, by game speed. The black line is the expected error rate for random movement.

between *latency and game speed* ($F_{12,228} = 3.34$, $p < .001$, $\eta^2 = .07$). The fastest (600ms) game speed 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 the 900ms game speed with error rate and latency (using neighboring points only) all show significant differences ($p < .01$) except for the following TTR pairings: 455:555 ($p = .02$), 655:755 ($p = .09$), and 755:855 ($p = .16$). For the 800ms speed, all neighbors were significantly different except 655:755 ($p = .14$). For the 700ms speed, all were different except 255:355 ($p = .88$) and 555:655 ($p = .88$). For the 600ms speed, only the 355:455 pairing was significantly different ($p < .01$).

There was greater variance in performance for the Space Invaders game than for study 2 (Pong). 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.

We also applied our Adjusted TTR measure to the performance data; as shown in Figure 6.4, ATTR again brings the curves for the different game speeds much closer together (although not as close as study 2). 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.

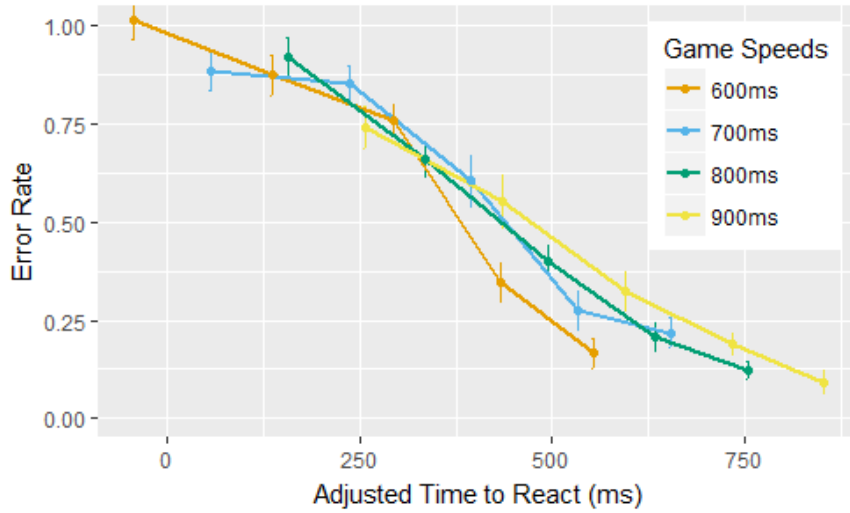


Figure 6.4: Error Rates with Adjusted TTR (Equation 5.1).

6.2.1 Player Accuracy Results

Player accuracy was calculated as the number of invaders hit divided by the total number of player shots for the round. *Latency* was not significantly correlated with accuracy (Pearson’s R of -0.02 , $p = .64$), and nor was *game speed* (R of 0.06 , $p = .24$). 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.

6.2.2 Experience Questionnaire Results

As in the Pong study from chapter 4, all survey questions were negatively correlated with latency (Figure 6.5); 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.

Although we asked fewer experience questions compared to the previous Pong study, game speed appears to have had less of an effect on player experience this time around. Kruskal-Wallis tests showed only showed a significance difference between *game speeds* on Q2: fun (Q1: $\chi^2 = 2.50$ and $p = .48$, Q2: $\chi^2 = 8.30$ and $p = .04$, Q3: $\chi^2 = 3.27$ and $p = .35$). Game speed only significantly affected how much fun was had (Q2), whilst not affecting perceived effectiveness (Q1) and attribution (Q3). It should be noted that game speed affected only the speed of enemy

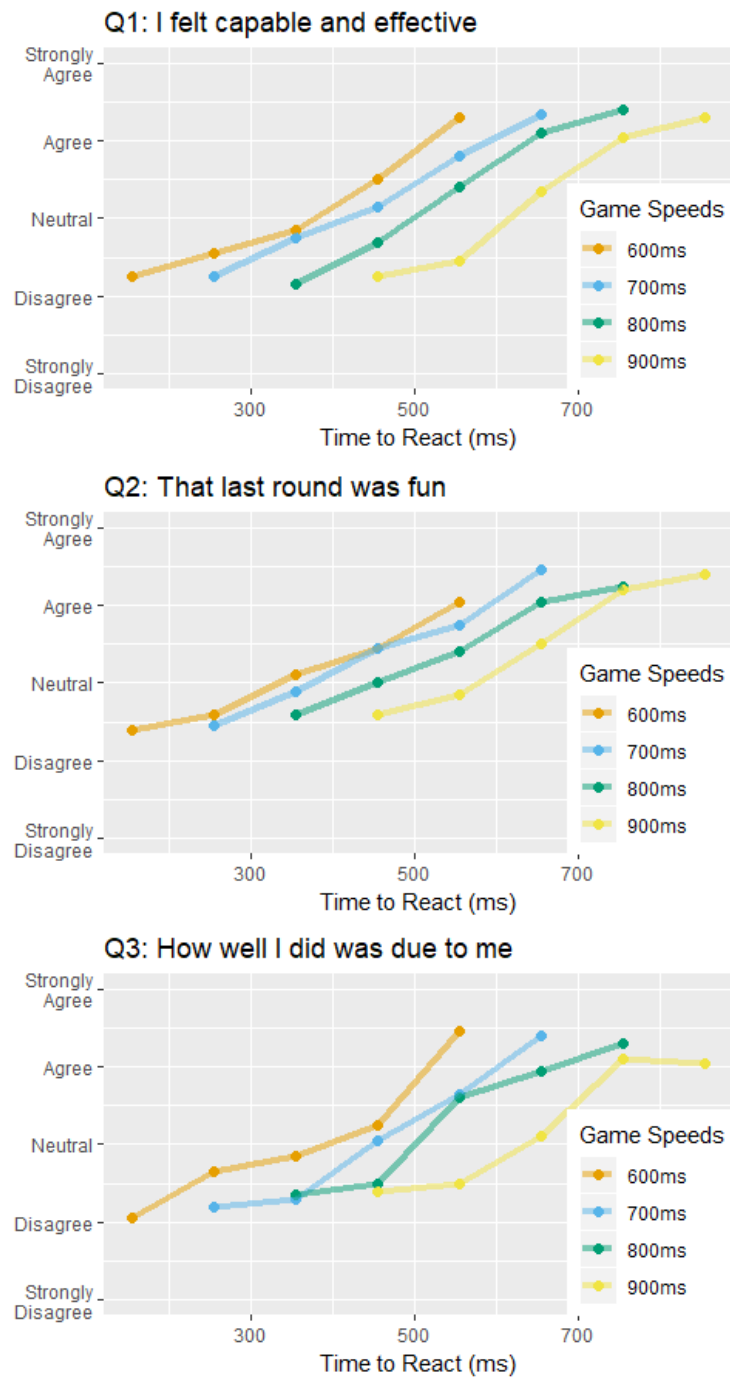


Figure 6.5: Responses to player experience questions (Table 6.1), vs. Time to React. Questions were scored on a 5-point Likert scale.

Game Speed	900ms	800ms	700ms	600ms
Mean Error Rate Difference	0.172	0.144	0.182	0.167

Table 6.3: Mean absolute differences between measured error rates and predicted error rates at different Game Speeds.

bullets, thus only affecting the dodging. It did not directly affect the shooting of enemies (though participants were told to focus on dodging). This difference could explain why players felt game speed affected them less, as only half the tasks were affected.

6.2.3 Evaluation of Predictive Model

We used the logistic function model developed in chapter 5 to predict error rates in this study. 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 $0.05 * 0.52 = 0.0125$. 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 6.6.

We calculated the goodness of fit of our model for Space Invaders 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 overall 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 study 2. 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.

Predicted error rates matched the measured error rates reasonably well when ATTR was low (left-hand side of Figure 6.6), 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 (Table 6.3) for different game speeds between 0.144 (800ms game speed) and 0.182 (700ms game speed).

Looking at Figure 6.6, 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

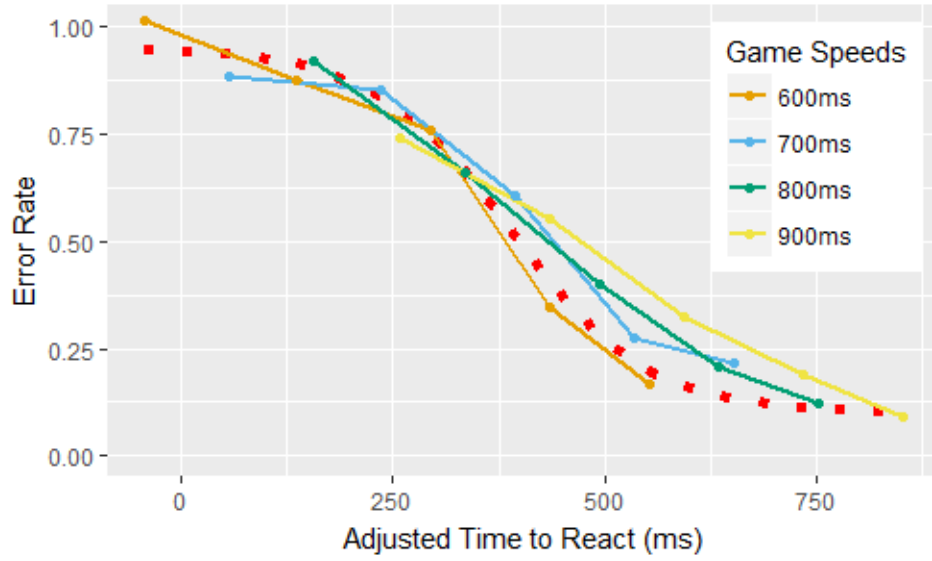


Figure 6.6: Error rates by game speed, with predicted error rate shown in red dots (calculated with Equation 5.2).

Adjusted Time to React	500ms	375ms	250ms
Measured Error Rate	0.367	0.591	0.802
Predicted Error Rate	0.270	0.562	0.849
Difference	0.097	0.029	-0.047

Table 6.4: 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 6.6).

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.

CHAPTER 7

DISCUSSION

In this chapter we discuss our study findings on latency, input devices, game atoms and error rate predictions along with various interactions. We discuss the implications of our findings for real world game designers, and examine whether our new knowledge may reduce time needed for latency-related play-testing.

This chapter is divided into five parts. We first provide a summary of our main findings. Second, we discuss results from the pointing devices and latency study. Third, we discuss results about game atoms — small, simple and repeatable tasks — from studies 2 and 3. Fourth, we discuss our predictive model. Lastly, we discuss the relevance of our findings outside the context of gaming.

7.1 Summary of Main Findings

Study 1: We examined four pointing devices (mouse, controller, drawing tablet, touchscreen) on both stationary and moving target acquisition tasks across multiple latency conditions. We recorded player performance (IP, Index of Performance from Fitts’ law), player experience, and cursor paths to calculate various path metrics suggested by Mackenzie et al. [51].

We found both latency and device to significantly affect IP. For stationary target acquisition, touchscreen performance was not affected by latency at all, and had by far the highest performance amongst all devices. The mouse also performed relatively well whilst both drawing tablet and controller had similarly poor performance. For moving target interception, the devices had the same overall performance rankings as in stationary target acquisition; the touchscreen was affected by latency for this task, but still had the highest overall performance. Having a moving target increased the task’s sensitivity to latency, and increased overall error rates.

Study 2: We examined 1D moving target interception — a small and simple game atom — under multiple latency conditions and game speeds in a custom version of Pong. Game speed determined the movement speed of the target ball. We recorded player performance (error rates)

and several player experience measures.

Game speed determined how sensitive the task was to latency, and we found both latency and game speed to significantly affect performance (number of errors). Since both game speed and latency had significant interactions, and since the different game speeds produced similar performance curves when plotted on a graph, we created a new measure called Time to React (TTR) that incorporated both terms. Player experience was also significantly affected by both latency and game speed.

Model and Study 3: We built a model capable of predicting error rates given game speed and latency using results from chapter 4. We found error rates from study 2 (chapter 4) to be shaped like a sigmoid curve when viewed with Adjusted TTR, and created our model using a modified logistic function to create similar curves. We found the model to have a good fit with our data.

Lastly, in chapter 6 we ran another study on a more complex task to validate our model. This study used a game similar to Space Invaders, and featured two goals: dodging and shooting. Again both game speed and latency significantly affected performance, with faster game speeds resulting in more errors. Our model fit the data relatively well, although work remains on fine-tuning the model's terms when multiple tasks are present.

7.2 Pointing Devices and Latency

7.2.1 Performance Explained by Cursor Paths

The path metrics suggested by Mackenzie et al. [51] allow for a more nuanced examination across pointing devices instead of merely examining pointing performance (though most path metrics were significantly correlated with IP — Index of Performance). Below we describe both cursor path metrics (visualized in Figure 7.1) and performance for each device used.

Controller: The controller had overall poor performance. This low performance can be explained in part by its cursor paths which generally moved in straight lines with few direction changes, with paths diverging from the optimal path as latency increased (seen in Figure 7.1). Users had great difficulty with clicking on the target when it was moving. This overall low pointing performance may be especially troublesome considering when a controller is often used — in conjunction with a TV that may have large amounts of display latency due to post-processing. Game designers should pay particular attention to the design of the game in these situations, as the controller's poor pointing performance may be compounded by latency.

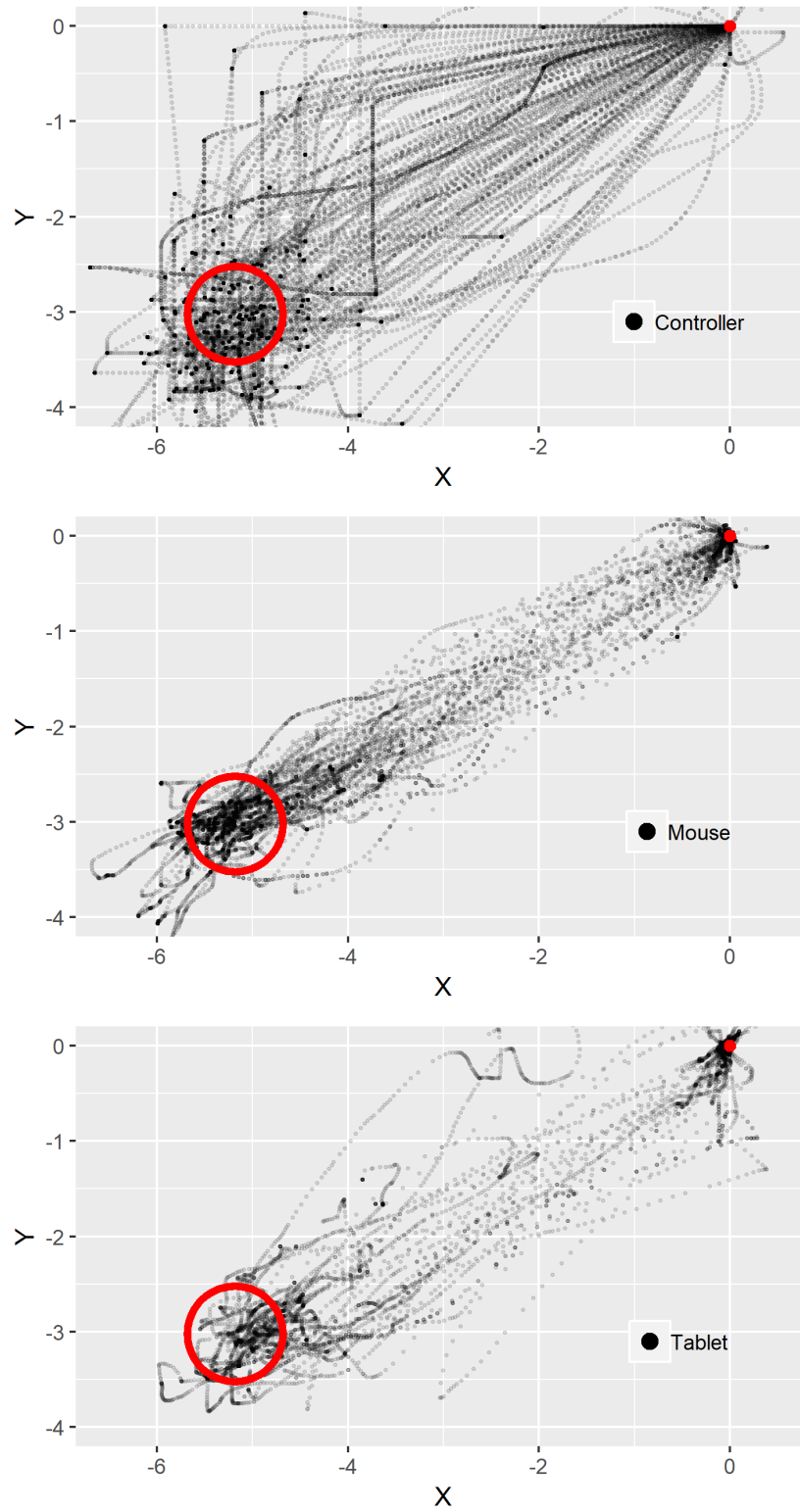


Figure 7.1: Random selection of cursor paths from controller, mouse and drawing tablet trials (reproduction of Figure 3.7 from study 1, chapter 3).

Drawing Tablet: The drawing tablet’s performance was roughly equivalent to the controller’s, with these two tied for lowest overall device performance. Again, cursor paths may help to explain this poor performance, with the drawing tablet producing erratic and inefficient cursor paths by straying considerably from the optimal straight-line path to the target (seen in Figure 7.1). The number of times the cursor went past the target (target overshoot) and the magnitude of overshoots increased with latency. This was likely caused by accidentally moving the pen when bringing it down for a tap.

Users were generally inexperienced with the drawing tablet, as evidenced by many of them periodically glancing at their hand while moving the pen, then glancing back up at the screen to see how far the cursor moved. This shows users were not confident in using the tablet as an absolute movement device. Despite our poor performance results for drawing tablets, however, it is not the case that drawing tablets always have poor pointing performance — there do exist experts who use drawing tablets in absolute movement mode for extremely fast-paced and highly-skill dependent games such as *Osu!* [60].

Mouse: The mouse had the highest pointing performance of all three indirect input devices (mouse, drawing tablet and controller), and was only beaten by the direct-input touchscreen. This good performance may be attributed to having the most efficient cursor paths that stayed relatively close to the optimal straight-line path. Mouse pointing performance was also found to be closely correlated with latency, with performance rapidly dropping as latency increased (though performance still stayed above that of controller and drawing tablet). It should be noted that expert mouse users that play fast-paced shooters sometimes use gaming mice that have adjustable DPI, and change their DPI for different situations (low DPI for sniping at long distance, high DPI for close-quarters shooting). Expert mouse users may have increased performance for our task if they were allowed to change their DPI settings mid-task.

7.2.2 Effects of Latency on Cursor Paths

The main effect of latency on cursor paths seen across all three indirect pointing devices (mouse, drawing tablet and controller) was the increase in necessary corrective actions such as changing cursor movement direction. Users had to correct their path more often to both arrive at the target and not overshoot it as latency increased. A pointing device capable of high performance in high latency conditions must be able to make minor course corrections, and have highly adjustable cursor movement speeds.

As seen in Table 3.3, of all cursor path metrics, target re-entry (TRE), was found to be most negatively correlated with performance (IP) across all latency levels. Target re-entry is where the cursor enters the target, and leaves again without clicking (sometimes multiple times). Moving the cursor past the target (target overshoot) can still happen without a target re-entry occurring if the cursor path is particularly wayward and never enters the target itself.

Following target re-entry, movement error (ME) and movement variability (MV) were the next two path metrics most negatively correlated with IP. These metrics measured how closely the optimal path to target was followed. Pointing devices must be capable of minute path adjustments in terms of both direction and speed to achieve low ME and MV. Minor adjustments in cursor movement direction are increasingly difficult with latency, as the outcome of moving the cursor is not seen until later. We also found ME, MV and TRE to be all highly correlated with each other.

7.2.3 Lessons for Designers

Based on the above findings, we created a number of design guidelines for various pointing devices. These findings focus on improving performance with the devices found with lower pointing performance (controller and drawing tablet). Many of the design guidelines focus on improving the critical cursor path metrics identified above. Improving these cursor path metrics should lead to increased pointing performance, both with and without latency.

Aim Assist for Drawing Tablets

For our studies the drawing tablet was set to absolute positioning mode, with each tablet corner being mapped to a corner of the screen. The cursor was moved by either hovering or dragging the pen along the tablet surface. To ‘click’ with the drawing tablet the user had to tap the tablet’s surface at a specific position with the pen. As can be seen in Figure 7.1, users had significant slippage (accidentally moving the pen) when bringing the pen down for a tap. Slippage led to many misclicks and target re-entries. Drawing tablet experts likely experience significantly less slippage when tapping with the pen, however our studies featured only novice users. To help novices, ‘sticky targets’ could be employed over virtual targets or on-screen buttons. Sticky targets slows cursor movement whilst over a target, thus reducing overshoot and making it harder for the cursor to leave the target due to minor movements [39].

Physical Buttons for Drawing Tablets

The negative effects of slippage are exacerbated when combined with local latency — it is difficult to make minor corrective actions when cursor movement is delayed. In order to reduce the negative effects of slippage, designers should try to minimize the number of pen taps. Physical buttons on the tablet, pen or keyboard could be used instead of virtual buttons to reduce necessary taps. This allows users to stay hovering above the surface, whilst using the pen-holding hand to click a button on the side of the pen, or using their other hand to click a button not situated not on the pen [32].

Wide Corridors for Controllers

The controller predominantly produced straight paths with few direction changes, with paths ‘widening’ and straying from the optimal straight-line path as latency increased (visualized in Figure 7.1). To take advantage of this phenomena, wide corridors with targets on fixed-angle paths from likely starting points could be implemented (e.g. the grid seen in Figure 7.2). Using wide corridors at 90° or 45° angles would make it easier to follow the optimal path, thus reducing the impact of movement error. It would also require fewer movement direction changes (MDC).

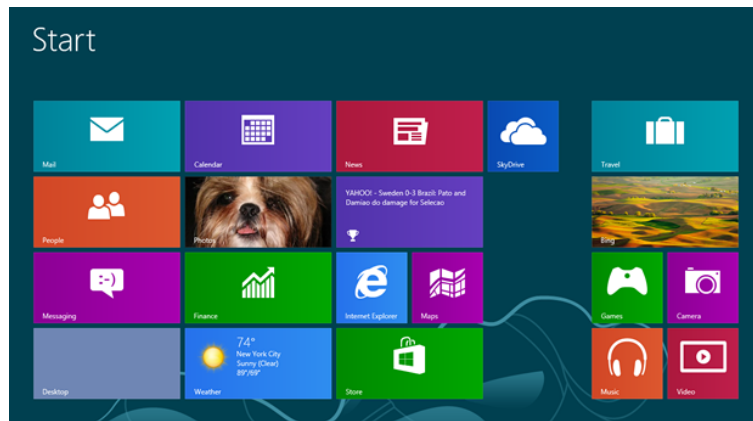


Figure 7.2: Example of large targets on fixed-angle paths (arranged in grid pattern) seen in Windows 8 Metro UI.

Target Magnetism for Controllers

Although already implemented in many console shooter games (e.g. Halo, Call of Duty) because of the inherent difficulty of aiming with a thumbstick, we recommend target magnetism for fast-paced pointing tasks involving controllers to improve resilience to local latency. Target magnetism makes

the cursor ‘lock on’ and moves the cursor to follow a moving target. This helps prevent target overshoot, and has been shown to increase performance under latency conditions in both stationary targeting and (especially) moving target interception tasks [39].

Use Touchscreens for Pointing Tasks

For both stationary target selection and moving target interception tasks, the touchscreen outperformed all other devices (though the mouse came relatively close for moving target interception). The direct input and absolute positioning of the touchscreen make it ideal for fast-paced pointing in the presence of local latency because the user’s finger is not affected by local latency. In indirect pointing devices, users rely on visual feedback to move the virtual cursor relative to its current position. This is not the case for touchscreens as they have direct input, and users need not wait for visual feedback from the system to select targets. 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 switching between relative movement with thumbsticks or absolute positioning using the touchscreen.

Low Latency has Significant Effects

Our study found total latencies of 110ms to have a significant effect on stationary target tasks, and total latencies of 60ms to significantly affect performance of moving target interception tasks. As other studies have shown [65, 39, 30, 31], low latencies are still noticeable by users. Care should be taken to reduce processing time and local latency as much as possible.

7.2.4 Relevance of Findings in the Real World

We now provide a summary of how this study and its data applies to the real world and aids game designers in designing better games. First, our cross-device performance comparisons saves game designers time when choosing a pointing device for their game. Configuring, performing and analyzing cross-device task performance play-tests can take substantial time. Our data and performance comparisons can aid game designers in making more informed decisions. We found higher performing devices to be correlated with improved player experiences, thus demonstrating the importance of choosing the best input device for the task.

Collecting and comparing cursor path metrics across multiple inputs devices is also useful in a variety of other contexts. For researchers, these cross-device studies are relatively rare, especially

under latency conditions. For game designers, knowing how each device differs in the act of pointing is useful for designing menus or other pointing interactions. When creating new input devices, it is useful to compare and contrast various input properties alongside cursor path metrics. In our study we determined target re-entry and movement error to be most significantly correlated with performance, thus informing input device designers about which cursor path metrics to focus on.

Last, after a game designer has chosen a specific input device for their game, our device-specific design recommendations provide a way to improve in-game player experience and performance with that device. These recommendations suggest ways to increase pointing effectiveness when using specific devices, whilst also making interactions more latency-resistant. In terms of applicability, pointing tasks are extremely common in both gaming and non-gaming applications, making these design guidelines widely applicable in various pointing contexts. More specifically, these recommendations are also well-suited to menu or option screen navigation, as menu screens feature many clickable targets.

7.3 Game Atoms, Game Speed and Latency

We will now discuss findings from our second and third studies (chapter 4 and chapter 6), as they share a number of similarities. Both studies examined game atoms, with the goal of determining how game speed and latency combine to affect performance and player experience. These studies focused on 1D pointing, with the third study featuring two simultaneous tasks (dodging and shooting) to help validate our model. Both studies measured error rates and player experience.

7.3.1 Game Speed Affects Performance and Experience

Game speed is the speed of interaction required by a game atom, and is the time taken to complete a single instance of the atom. The game speed for Pong was measured as the time it took the ball to travel from the top of the screen to the player's paddle. For Space Invaders, game speed was the length of time it took an enemy bullet to move from the top of the screen to the player. Game speed and added local latency for both studies were varied to see their effect on performance and experience.

Overall, game speed affected how sensitive performance was to local latency. Both game speed and latency are closely related: a faster game speed increases latency's effects, thus causing latency to have a larger negative impact on error rates. This effect can be seen in Figure 4.3, which compares

the error rates of different game speeds across latency levels. This relationship between latency and game speed makes sense — a faster game speed reduces the available time for the user to process and react to an event, and latency further lowers this available time. This relationship was found in both studies (study 2 and 3). Error rates were found to be significantly correlated to game speed, with both game speed and latency also having significant interaction effects. This demonstrates that no conversation about latency is complete without mentioning game speed.

This relationship means that reducing game speed may be an effective tool for reducing error rates. Changing the game speed alters how resilient the task is to latency, and reducing game speed will reduce errors due to latency. However, slowing the game speed too much may result in gameplay that is too easy, also affecting player experience.

Our player experience results (Figures 4.6 and 6.5) from both Pong (study 2) and Space Invaders (study 3) were somewhat conflicting. For study 2 (Pong), game speed significantly affected most player experience questions (capability and effectiveness, fun, effort invested, and frustration), whilst for study 3 (Space Invaders) it only affected one out of three experience measures (fun). As previously noted, game speed in Space Invaders only affected how difficult it was to dodge enemy bullets and did not effect shooting enemies. This may have impacted player experience results, as only one of the two objectives was getting harder (dodging). Although more work must be done in this area, we believe that game speed will significantly effect most player experience measures for most tasks.

7.3.2 Time to React

To better view the effects of both latency and game speed on error rate, we created a novel measure called Time to React. Time to React (TTR) is how much time the user has to react before an error could occur. It is a function of both game speed and latency, as faster game speeds and higher levels of latency reduce the available time to react. Time to React accounts for local latency, as it is calculated as game speed minus latency — a successful action has to be initiated *before* Time to React reaches 0, else the delay from latency will make the action occur too late to prevent an error. By filtering out the latency from the measure we can view latency’s effects more easily.

We created an improved version Time to React called Adjusted Time to React (Equation 5.1), that weighs latency more heavily as it increases. Adjusted Time to React grouped our error rate performance curves more closely together, allowing them to be modelled using a single sigmoid curve equation.

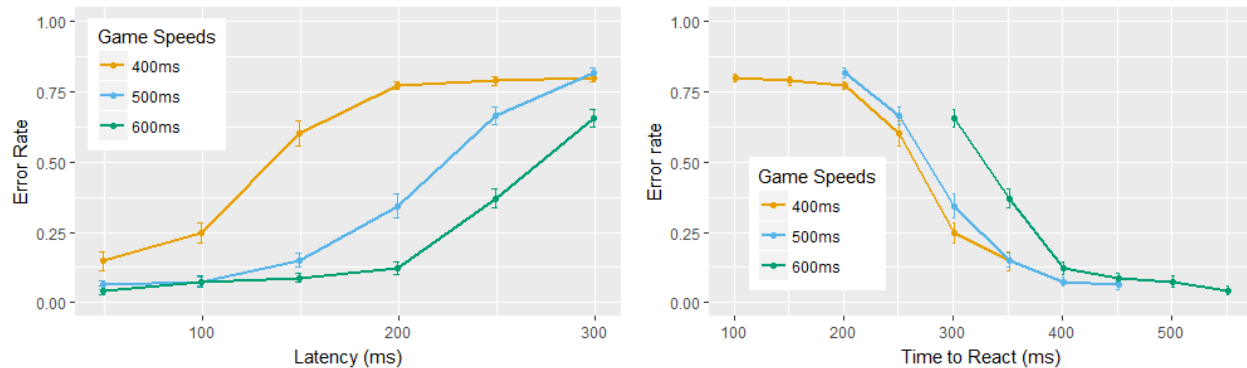


Figure 7.3: Left: Error rates of multiple game speeds with latency on X axis. Right: Error rates of multiple game speeds with Time to React on X axis. Both figures using results from study 2 (reproduction of Figures 4.3 and 4.4 from chapter 4).

We believe that Adjusted Time to React (ATTR) is a useful measure for game designers for several reasons. First, using ATTR as one axis of a graph allows for a proper comparison between multiple game speeds and latencies. Without using ATTR, it is difficult to visualize how latency actually affects the task (this can be seen in Figure 7.3). Second, Adjusted Time to React is also easy to calculate, and thus should be suitable for both game designers and researchers. Third, understanding the relationship between game speed and latency is an important lesson for designers when creating game atoms. We suggest creating similar figures to the right side of Figure 7.3 when testing core interactions, as this will provide some insight into the possible range of game speeds for future play-testing.

Although similar to Claypool’s proposed Deadline measure [12], our Time to React measure differs in that it is used directly for predictive modelling, whereas Claypool treats Deadline as an abstract concept for categorising game actions. We give an equation for calculating ATTR (Equation 5.1), that can be used in other studies when analyzing latency’s effects alongside interaction speed.

7.3.3 Latency’s Effects Are Exponential

Previous studies have suggested that latency’s effects are non-linear. Mackenzie and Ware suggested a multiplicative term for latency in Fitts’ law [53, 77]. Claypool adapted Fitts’ law to have an exponential latency term when dealing with moving targets after comparing data from mouse and gamepad thumbstick interception tasks [11, 8].

As seen in Figure 7.4), the right-hand side of our performance curves adopt exponential growth up until the middle inflection point of the sigmoid curve. This beginning transition area is crucially important — errors due to latency begin to rise rapidly, and game designers should make sure their

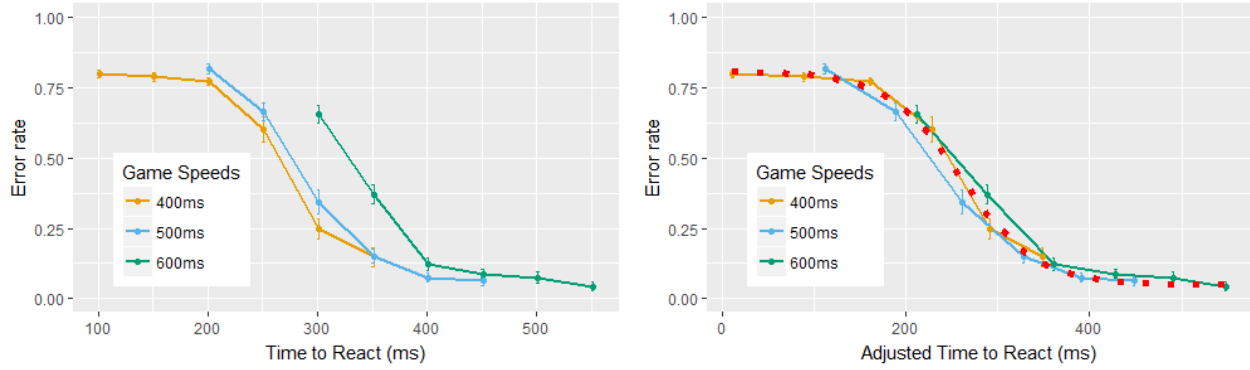


Figure 7.4: Left: Error rates with a linear term for Time to React. Right: Error rates with quadratic term for TTR. Both figures using results from study 2 (reproduction of Figures 4.4 and 5.2 from chapter 4).

game atoms are positioned further to the right of this area. Further to the left of the central sigmoid inflection point, error rate growth slows and finally settles into an error rate ceiling. This growth is no longer exponential, however this area is less important as the beginning transition area, as many players will have already given since errors due to the high amount of errors due to latency.

7.4 Predictive Error Rate Model

By examining results from study 2 (Pong) in chapter 4, we created a model that takes Time to React (local latency and game speed) and other parameters to predict errors in a game atom. Below we discuss our model and its associated findings, and detail how it be generalized and applied in the real world.

7.4.1 Error Rates Followed Logistic Functions

Error rates from both studies 2 and 3 followed S-shaped sigmoid curves (more so in study 2). As seen in Figure 7.5, these curves have several identifiable features: a clearly delineated floor that has the lowest values, a ceiling featuring the highest values, and a critical transition area between floor and ceiling. The beginning and end positions of the transition area are important for determining critical local latency thresholds for when latency's effects begin to take significant and negative effect (pictured in red in Figure 7.5). Game designers should try to ensure that expected levels of local latency are low enough to not be within the transition area. The steepness of the transition area is also important, as the steepness is determined by both game speed and latency — the faster the game speed and higher the latency, the steeper the slope will be. The slope defines how sensitive

the atom is to changes in Time to React (game speed and latency).

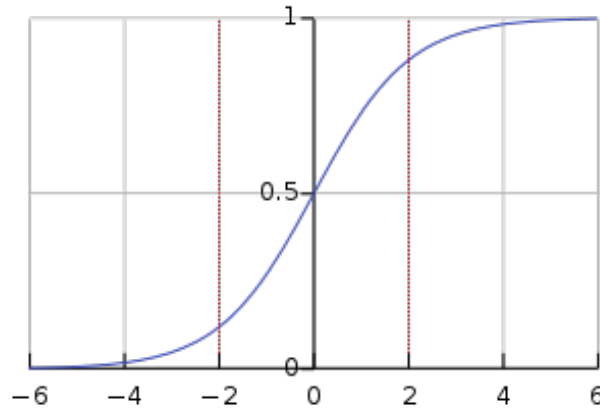


Figure 7.5: Example sigmoid curve featuring a floor (left), transition area (denoted in red), and ceiling (right). Reproduced from Figure 2.8 in chapter 2.

Using logistic functions to predict error rates is a new approach in game design, however they have been extensively used in other fields such as psychology as part of psychometric functions. For example, psychometric functions are used to model stimuli thresholds of human perception. In these situations participants have a binary response: yes (the change in stimulus was noticeable), or no (it was not). This lends itself well to binary errors: either an error occurred, or it did not. When plotted, psychometric functions measure the percent chance that the participant will respond yes or no (on the Y axis), and the stimulus intensity of the (X axis). We translated this percent chance on the Y axis into a normalized error rate, going from 0 to 1, where 0 meant no errors were made, and 1 that every possible error was made. The X axis charted our version of stimulus intensity, which was Time to React.

7.4.2 Usefulness of the Model

For our model to be of actual use to game designers, it must accomplish three objectives: the model must be easy to use, easy to understand, and must provide tangible benefits.

Ease of use is important, as using the model must take less time than simply performing additional latency-related play-testing. We believe our model to be easy to use, as its terms are clearly defined and easily understood. Terms such as the floor and ceiling of the logistic function are filled by simple equations. We also use easy-to-determine variables such as game speed and number of tasks. It is easy to test multiple game speeds, as only the Time to React needs to be recalculated

(and other steps to apply model remain the same). Step-by-step instructions for using the model are provided at the end of chapter 5.

As for ease of understanding, error rate is a measure that should be easily understood. Error rate is the percentage of errors relative to successes (non-errors) when repeatedly performing a task. Errors are undesirable, and having too high of an error rate will frustrate players. Our model supplies an error rate which game designers can error rate to judge if the number of user errors due to latency are acceptable or not.

Calculating an error rate could be as simple as the ratio of failures to successfully completed trials (e.g. the number of times the player missed the ball in Pong divided by total number of possible hits). Determining how to calculate the upper limit of an error rate is highly task-dependent, and can be complex if both time and random elements are added (see error rate calculation for Space Invaders in chapter 5). We feel that our error rate measure is as intuitive as it can be to both calculate and understand the phenomena.

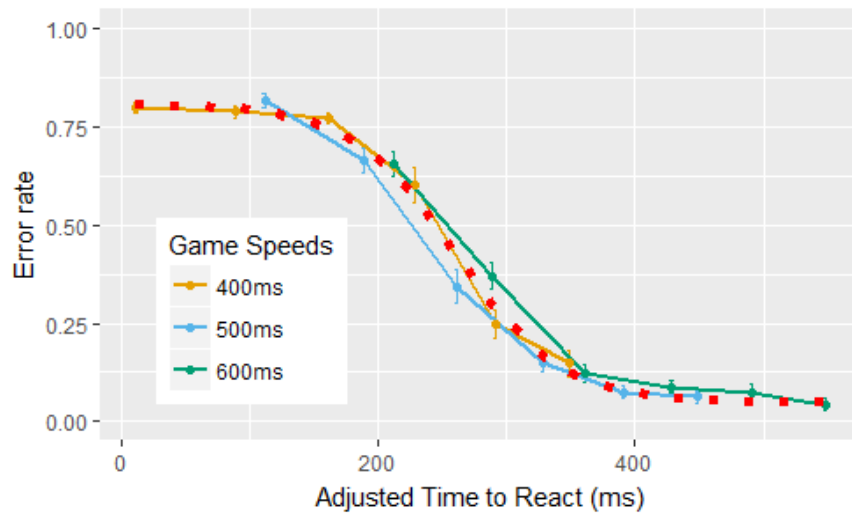


Figure 7.6: Error rates from study 2 (chapter 4), with model-predicted error rates in red. Reproduced from Figure 5.2 in chapter 4.

In addition to understanding how error rates are calculated and what error rates are, game designers must also understand how to examine error rate graphs. When examining a graph such as Figure 7.6, the critical transition area between floor and ceiling can be seen (from roughly 190ms TTR to 330ms TTR). Designers should aim for their interaction to be somewhere to the right of the center inflection point so as to increase the interaction’s latency resistance if latency were to increase due to varying gaming setups. The desired level of errors is also up to the game designer, as that

determines the challenge level or difficulty of the interaction, and an interaction that is too easy is likely undesired. Task difficulty could also be adjusted dynamically based on player performance — the game speed is slowed when the player is performing poorly (which may be caused by high amounts of local latency from their gaming setup).

One of the main benefits of our predictive model is that it reduces latency-related play-testing. Considerable time is currently spent on play-testing, and being able to reduce play-testing saves both time and money. With our easy-to-use model, game designers will not have to invest as much time in latency-related play-testing. Examples of latency-related play-testing would be repeatedly testing core interactions using different input devices, monitors and CPU's to achieve different amounts of local latency. Our model shows the effects of both latency and game speed on interactions, and can thus identify interactions that are sensitive to latency. It is also easy to check the effects of multiple game speeds.

Despite making our model easy to use in many aspects, it should be noted that using our model still requires considerable calibration and carefully specifying game atoms. There remains work to be done in simplifying the model.

7.5 Applications Outside Gaming

Our research is primarily aimed at gaming, since gaming features many fast-paced tasks that require high amounts of precision. However, our findings can also be applied in other contexts outside of gaming. Firstly, study 1 examines regular pointing with various input devices. Pointing is a common task that is performed in many computing tasks, including usage of most desktop operating systems. Local latency will still be present when using non-gaming software, and our cursor path analysis will be relevant. Although the effect of an error occurring during non-gaming software usage is arguably less severe than when playing games, any pointing performance improvements under latency conditions would be quite beneficial.

For a more extreme example, remote operation of heavy equipment is likely subject to substantial amounts of latency. When operated remotely, control will be done via cameras and sensors, and comparisons can be made with videogames. The best input methods need to be chosen, and our comparison of input devices may be useful. Errors with heavy equipment can be both dangerous and costly, and error rates of performing common tasks with heavy equipment need to be carefully studied. Our predictive model may show designers what levels of latency are acceptable for remote

operation of heavy equipment, and simulation done ahead of time to identify latency-sensitive tasks will improve safety and save money.

7.6 Limitations

In this work, we evaluated latency and game speed's effect on input devices, player performance and player experience in videogames. We performed multiple user studies where participants played games across various latency conditions. Although our results are an important first step towards understanding local latency's effects, our approach and implementation led to some limitations.

7.6.1 Study Design

Limited time restricted the number of conditions we could study. For study 1 (input device study, chapter 3) we only had time to study four pointing devices (mouse, controller, touchscreen, drawing tablet). These pointing devices captured several important combinations of pointing device characteristics, but it did not test a direct input, indirect movement device such as virtual 'soft' joysticks on a touchscreen (Figure 2.3).

We also would have liked to include a dragging task, in addition to the stationary and target acquisition tasks. For both study 2 and 3 (Pong in chapter 4 and Space Invaders in chapter 6), adding more latency conditions would expand both the range of analysis and provide finer granularity. Examples include adding a 300ms and 350ms latency levels to study 2, and having latency levels every 50ms instead of 100ms for study 3. Unfortunately, we found that participants tired quickly when the experiment went over 20 minutes, thus restricting the number of tasks and conditions.

Another limitation in all of our studies was the prior experience of participants with input devices and videogames. When recruiting participants, the only restriction was that they have at least passing familiarity with a mouse. Once the participant arrived at the study itself, they self-reported their expertise with the input device being used along with average weekly time spent playing videogames. This meant that some users had far more prior device experience than others. This was especially prevalent in study 1 (chapter 3), where only a few participants had used a drawing tablet before. It is possible the lack of experts for each device may have influenced our results, as we did not reject any participants due to prior expertise with input devices.

In future a covariate analysis could be performed with player skill/experience with each device added as a covariate. Alternatively, more care could be taken to guarantee the recruitment of a

spread of skill sets with each input device.

7.6.2 Latency Simulation

Since we are studying latency and its effects, it is important to carefully examine the method by which we simulate latency. For all of our studies, additional artificial latency was simulated by buffering all received input and later using it after an elapsed period of time. With this method, the exact amount of delay could be chosen. The game's framerate was set to a consistent and known rate (120fps), with an internal fixed update period of 1ms for polling input. This approach simulated additional latency at the input device level, as if the input device polling was delayed by a certain amount of time. Local latency is the total delay from the many steps from input to output.

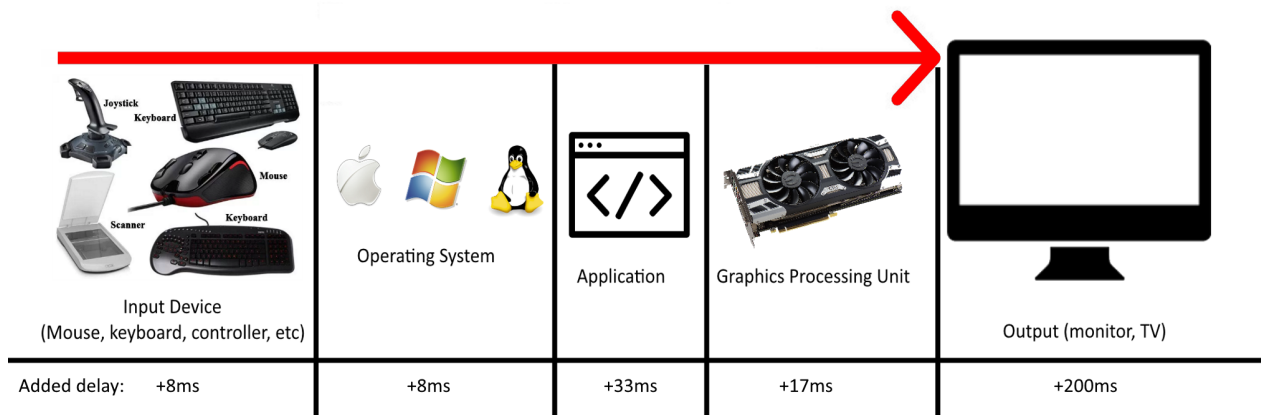


Figure 7.7: Each step in the process of gathering input to displaying output adds delay, with example amounts of delay below each step. Note the high 200ms latency from the monitor.

It is possible however, for large amounts of delay to occur at different points in this process, thus changing the game outcome. Consider the example pictured in Figure 7.7: a monitor displaying the game performs post-processing, thus delaying the images seen. However, input is not very delayed, thus allowing clicks to register quickly, and a task may be successfully completed even though the display has yet to be updated. Changing when the clicks are registered may affect player performance in moving-target interception tasks.

Another factor to consider is that our approach does not emulate any jitter (variation in latency). However, 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 latency (input devices, software

processing, and display devices) are typically much less variable. This means our work may not translate as well to networked applications that have jitter.

An additional limitation in our studies is the baseline system latency itself. Every system has an inherent amount of local latency. This was measured in each of our studies by examining frame-by-frame footage from a high-speed camera recording gameplay. This general amount of base system latency (around 50ms) is unavoidable in current consumer computers, and is our ‘best case’ of lowest measured latency (no extra latency added). Previous studies have shown users to perceive latencies as low as 2ms, and have their performance affected at 25ms of latency [1, 42]. Special hardware and projectors were used to achieve such low latencies in these studies. Ideally we would examine these lower latencies to provide a more comprehensive view on latency’s effects; however, we know that lower latencies in our studies (e.g 50-100ms total latency) had little effect on player performance and experience for our tasks. Thus we believe that examining lower latencies would simply confirm low amounts of latency to have very little overall effect. We believe it to be most valuable to study latency values currently encountered by average users.

7.7 Future Work

The above limitations represent opportunities yet to be explored that may be expanded upon in future work.

7.7.1 Hybrid Input Devices

There many different kinds of input devices, all with varying properties. In chapter 3 we studied four pointing devices (mouse, controller, touchscreen and drawing tablet), however in future we would like to study additional devices. Of particular note are indirect input and relative movement devices such as virtual ‘soft’ joysticks on touchscreens. Additionally, many input devices combine several techniques and methods for acquiring input, sometimes allowing the user to choose their preferred method. Below we list examples of current hybrid input devices.

- The Wii U’s gamepad includes traditional thumbsticks and buttons whilst incorporating a touchscreen.
- The Nintendo 3DS has a touchscreen along with directional pad and buttons.
- Both the Steam and PS4 controller include a touchpad (but no screen).

- VR headsets can now track gaze, which could potentially be used for pointing.

Studies could be performed to test which input ‘method’ of a hybrid device is preferred by users when latency is applied. This would allow for the construction of more latency-resistant input devices, with the option of switching control modes for various tasks and levels of latency.

7.7.2 Predicting Player Experience

In chapter 4, we found player experience curves to resemble the sigmoid curves seen when analyzing error rates. It may be possible to predict player experience using an adapted version of our predictive model. Modelling player experience would allow designers to evaluate the costs and benefits of lowering overall latency in terms of player experience. Although player performance is important, in the end, games are meant to create experiences for players, thus making experience extremely important. A cost-benefit analysis could be performed to see the effects of frame-rate and gaming setup have on player experience.

7.7.3 Jitter

As previously mentioned, our added artificial latency did not simulate any jitter. Our studies could be repeated with jitter applied to the added artificial applied latency, with real world patterns and profiles of jitter from network latency being applied. This would further extend and test our model for predicting error rates into the realm of networked games. This, however, would be a difficult task — predicting error rates with networked games will be even more complex, as errors will be affected by any latency compensation or consistency maintenance techniques that are often implemented in multiplayer games.

7.7.4 Regression and Data-Driven Parameters

Parameters for our predictive model were chosen and tested by hand. This exploratory approach allowed us to explore and gain new understanding of the requirements necessary to model error rates of a given task. An alternative approach would be to use a non-linear regression to fit our parameters and performance curve. However, this may lead to overfitting, and make it difficult to generalize the model to other tasks.

Another data-driven approach would be to have users perform play-testing, plot all of their performances, and use non-linear regression to fit the curve. This would lead to an accurate model for

one task, but involves performing a time-consuming and costly study (and our model was originally created to avoid such costly studies). However, this approach may still be worth investigating to see if our sigmoid curves persist across many game genres.

7.7.5 Error Rate Model Validation

Lastly, our error rate prediction model could use additional validation on various tasks. We only tested it on interception pointing tasks with a mouse. A catalogue of game atoms needs to be created, including atoms such as aiming in shooters, jumping in a platformer, and blocking in fighting games. These atoms should be further analyzed and broken down into more abstract terms such as interception, targeting, key-press sequences that a generic formulation of the model can be applied to.

Each task will require calibration using different game speeds, as each task operates on a unique timescale. Additionally, reasonable latency values based on setting and intended real world must be determined. Additional validation and refinement will make our model more useful to game designers for real world tasks.

CHAPTER 8

CONCLUSION

In this thesis we set out to address the lack of knowledge of how videogames are affected by local latency. Our solution came in three parts: providing information on the effects of local latency on input devices, providing information on how game atoms are affected by both game speed and latency, and by building a model that predicts error rate given game speed and latency values.

For the effects of latency on input devices, our contributions include a cross-device comparison of four popular gaming pointing devices: controller, mouse, touchscreen and drawing tablet. We analyzed both performance and player experience, as well as cursor path metrics. These results will help game designers in choosing a pointing device for their game and provide additional information to input device designers to help build better and more latency-resistant input devices. Our device-specific guidelines will also help improve pointing performance in both gaming and non-gaming applications.

To understand local latency’s effects on game atoms, we performed a study featuring 1D target interception with varying levels of local latency and game speed. Based on results from this study we created new term called ‘Time to React’ to help view latency’s effects on performance. We also found latency’s effects on performance to be exponential, helping to confirm previous research.

Our last major contribution is to provide a predictive error rate model. This model allows game designers to quickly identify latency-sensitive game atoms while also comparing the effects of multiple game speeds. This can save valuable time play-testing, as game designers can focus on these problematic atoms and adjust their designs accordingly. We outline the required steps to use our model, and suggest how to use each term in the equation. We also performed an additional study on a more complex task to help validate our model.

This thesis also makes a number of minor contributions:

- In study 1, controls and calibration for four different pointing devices had to be performed. The indirect input and relative movement of the gamepad required much play-testing to calibrate proper cursor speeds and thumbstick dead zones.

- We adapted both Pong and Space Invaders (studies 2 and 3) to be suitable for studies. We provide information on these reference implementations for future use and study.
- We also identified a number of different game atoms (though we did not study all of them).
- We provide a useful link between psychometric/logistic functions used in psychology to determine just-noticeable differences between stimuli and binary error rates from player performance in videogames.

We hope that game designers will find our contributions useful and use them to save both time and money in play-testing and to design better games. Many fast-paced game genres can suffer from local latency, and having a ‘responsive’ game ranks highly among desired attributes for a game, making this work widely applicable. Additionally, with renewed interest in cloud gaming services that have traditionally suffered from both local and network latency, examining latency and its effects have never been more relevant. Google, Amazon, Sony and Microsoft have all announced or have already started implementing their own cloud gaming services and must invest resources to both study and reduce latency’s effects on their services.

REFERENCES

- [1] Glen Anderson, Rina Doherty, and Subhashini Ganapathy. User Perception of Touch Screen Latency. In Aaron Marcus, editor, *Design, User Experience, and Usability. Theory, Methods, Tools and Practice: First International Conference, DUXU 2011, Held as Part of HCI International 2011, Orlando, FL, USA, July 9-14, 2011, Proceedings, Part I*, pages 195–202. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.
- [2] Apple thunderbolt display - technical specifications. https://support.apple.com/kb/sp642?locale=en_CA. Accessed: 2017-8-5.
- [3] Rog swift pg278q — monitors — asus canada. https://www.asus.com/ca-en/Monitors/ROG_SWIFT_PG278Q/specifications/. Accessed: 2017-8-5.
- [4] Tom Beigbeder, Rory Coughlan, Corey Lusher, John Plunkett, Emmanuel Agu, and Mark Claypool. 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*, pages 144–151, New York, NY, USA, 2004. ACM.
- [5] Ugo Braga Sangiorgi, Vivian Genaro Motti, François Beuven, and Jean Vanderdonckt. Assessing Lag Perception in Electronic Sketching. In *Proceedings of the 7th Nordic Conference on Human-Computer Interaction: Making Sense Through Design*, NordiCHI '12, pages 153–161, New York, NY, USA, 2012. ACM.
- [6] Stuart K Card, Thomas P Moran, and Allen Newell. The keystroke-level model for user performance time with interactive systems. *Communications of the ACM*, 23(7):396–410, 1980.
- [7] Kajal T. Claypool and Mark Claypool. On frame rate and player performance in first person shooter games. *Multimedia Syst.*, 13(1):3–17, September 2007.
- [8] M Claypool. On Models for Game Input with Delay; Moving Target Selection with a Mouse. In *2016 IEEE International Symposium on Multimedia (ISM)*, pages 575–582, dec 2016.
- [9] M. Claypool and D. Finkel. The effects of latency on player performance in cloud-based games. In *2014 13th Annual Workshop on Network and Systems Support for Games*, pages 1–6, Dec 2014.
- [10] Mark Claypool. The effect of latency on user performance in real-time strategy games. *Computer Networks*, 49(1):52 – 70, 2005. Networking Issue in Entertainment Computing.
- [11] Mark Claypool. Game Input with Delay; Moving Target Selection with a Game Controller Thumbstick. *ACM Trans. Multimedia Comput. Commun. Appl.*, 14(3s):57:1—57:22, jun 2018.
- [12] Mark Claypool and Kajal Claypool. Latency and player actions in online games. *Commun. ACM*, 49(11):40–45, November 2006.

- [13] Mark Claypool and Kajal Claypool. Latency can kill. *Proceedings of the first annual ACM SIGMM conference on Multimedia systems - MMSys '10*, page 215, 2010.
- [14] Mark Claypool, Ragnhild Eg, and Kjetil Raaen. 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*, pages 117–123, 2016.
- [15] Shadow and the future of cloud computing. https://www.xtof.info/blog/wp-content/uploads/2016/08/Cloud_Gaming-640x249.png, August 2016. Accessed: 2019-03-18.
- [16] Ihs markit report: Cloud service providers make powerplay in revitalized cloud gaming market. <https://technology.ihs.com/611863/ihs-markit-report-cloud-service-providers-make-powerplay-in-revitalized-cloud-gaming-market>, March 2019. Accessed: 2019-03-18.
- [17] Andy Cockburn, Carl Gutwin, and Saul Greenberg. A predictive model of menu performance. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '07, pages 627–636, New York, NY, USA, 2007. ACM.
- [18] Controller input lag comparison. https://www.reddit.com/r/RocketLeague/comments/8wvf9u/controller_input_lag_comparison_more_info_in_the/e1yor20/. Accessed: 2018-10-27.
- [19] Ansgar E Depping and Regan L Mandryk. 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, pages 1040–1052, New York, NY, USA, 2017. ACM.
- [20] Matthias Dick, Oliver Wellnitz, and Lars Wolf. 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, pages 1–7, New York, NY, USA, 2005. ACM.
- [21] Digby Elliott, Werner F Helsen, and Romeo Chua. A century later: Woodworth's (1899) two-component model of goal-directed aiming. *Psychological bulletin*, 127(3):342, 2001.
- [22] With viewership and revenue booming, esports set to compete with traditional sports. <https://onlinebusiness.syr.edu/blog/esports-to-compete-with-traditional-sports/>, January 2019. Accessed: 2019-03-07.
- [23] Paul M Fitts. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of experimental psychology*, 47(6):381, 1954.
- [24] Ira Flatow. This month in physics history: October 1958: Physicist invents first video game. *American Physical Studies News*, 17(9):151–168, October 2008.
- [25] Carl Gutwin. The Effects of Network Delays on Group Work in Real-Time Groupware. In Wolfgang Prinz, Matthias Jarke, Yvonne Rogers, Kjeld Schmidt, and Volker Wulf, editors, *ECSCW 2001: Proceedings of the Seventh European Conference on Computer Supported Cooperative Work 16–20 September 2001, Bonn, Germany*, pages 299–318. Springer Netherlands, Dordrecht, 2001.

- [26] Carl Gutwin, Steve Benford, Jeff Dyck, Mike Fraser, Ivan Vaghi, and Chris Greenhalgh. Revealing Delay in Collaborative Environments. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04, pages 503–510, New York, NY, USA, 2004. ACM.
- [27] Sandra G. Hart. Nasa-task load index (nasa-tlx); 20 years later. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 50(9):904–908, 2006.
- [28] Sandra G Hart and Lowell E Staveland. Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research. In Peter A Hancock and Najmedin Meshkati, editors, *Human Mental Workload*, volume 52 of *Advances in Psychology*, pages 139–183. North-Holland, 1988.
- [29] Tristan Henderson and Saleem Bhatti. Networked games: A qos-sensitive application for qos-insensitive users? In *Proceedings of the ACM SIGCOMM Workshop on Revisiting IP QoS: What Have We Learned, Why Do We Care?*, RIPQoS '03, pages 141–147, New York, NY, USA, 2003. ACM.
- [30] Niels Henze, Markus Funk, and Alireza Sahami Shirazi. Software-reduced touchscreen latency. *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services - MobileHCI '16*, pages 434–441, 2016.
- [31] Niels Henze, Sven Mayer, Huy Viet Le, and Valentin Schwind. Improving Software-reduced Touchscreen Latency. In *Proceedings of the 19th International Conference on Human-Computer Interaction with Mobile Devices and Services*, MobileHCI '17, pages 107:1—107:8, New York, NY, USA, 2017. ACM.
- [32] Ken Hinckley, Patrick Baudisch, Gonzalo Ramos, and Francois Guimbretiere. Design and analysis of delimiters for selection-action pen gesture phrases in scriboli. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '05, pages 451–460, New York, NY, USA, 2005. ACM.
- [33] Ken Hinckley and Daniel Wigdor. Input technologies and techniques. *The human-computer interaction handbook: fundamentals, evolving technologies and emerging applications*, pages 151–168, 2002.
- [34] Errol R Hoffman. Fitts' Law with transmission delay. *Ergonomics*, 35(1):37–48, 1992.
- [35] How the nintendo nes zapper worked, and why it doesn't work on hdtvs. <https://www.howtogeek.com/181303/htg-explains-how-the-nintendo-zapper-worked-and-why-it-doesnt-work-on-new-tvs/>, October 2018. Accessed: 2019-04-01.
- [36] Human Benchmark - Reaction Time Stats. <https://www.humanbenchmark.com/tests/reactiontime>.
- [37] Ray Hyman. Stimulus information as a determinant of reaction time. *Journal of experimental psychology*, 45(3):188, 1953.
- [38] Input lag of tvs. <https://www.rtings.com/tv/tests/inputs/input-lag>, February 2019. Accessed: 2019-03-07.

- [39] Zenja Ivkovic, Ian Stavness, Carl Gutwin, and Steven Sutcliffe. 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*, 1:135–144, 2015.
- [40] Richard J Jagacinski, Daniel W Repperger, Sharon L Ward, and Martin S Moran. A Test of Fitts' Law with Moving Targets. *Human Factors*, 22(2):225–233, 1980.
- [41] Bonnie E. John and David E. Kieras. The goms family of user interface analysis techniques: Comparison and contrast. *ACM Trans. Comput.-Hum. Interact.*, 3(4):320–351, December 1996.
- [42] Ricardo Jota, Albert Ng, Paul Dietz, and Daniel Wigdor. 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, pages 2291–2300, New York, NY, USA, 2013. ACM.
- [43] Harold H Kelley. Attribution theory in social psychology. In *Nebraska symposium on motivation*. University of Nebraska Press, 1967.
- [44] Shawn Knight. Playstation and Xbox controllers most popular among Steam gamers - Techspot. <https://www.techspot.com/news/76631-playstation-xbox-controllers-most-popular-among-steam-gamers.html>, 2018. Accessed: 2019-1-9.
- [45] Yeng-Ting Lee, Kuan-Ta Chen, Han-I Su, and Chin-Laung Lei. Are all games equally cloud-gaming-friendly?: An electromyographic approach. In *Proceedings of the 11th Annual Workshop on Network and Systems Support for Games*, NetGames '12, pages 3:1–3:6, Piscataway, NJ, USA, 2012. IEEE Press.
- [46] Hui Li, Ying Liu, Jun Liu, Xia Wang, Yujiang Li, and Pei-Luen Patrick Rau. Extended klm for mobile phone interaction: A user study result. In *CHI '10 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '10, pages 3517–3522, New York, NY, USA, 2010. ACM.
- [47] Lpe: The videogames' industry is bigger than hollywood. <https://lpesports.com/e-sports-news/the-video-games-industry-is-bigger-than-hollywood>, October 2018. Accessed: 2019-03-07.
- [48] Mitchell JH Lum, Jacob Rosen, Hawkeye King, Diana CW Friedman, Thomas S Lendvay, Andrew S Wright, Mika N Sinanan, and Blake Hannaford. Teleoperation in surgical robotics—network latency effects on surgical performance. In *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE*, pages 6860–6863. IEEE, 2009.
- [49] I. Scott MacKenzie. Fitts' law as a research and design tool in human-computer interaction. *Human-computer interaction*, 7(1):91–139, 1992.
- [50] I. Scott MacKenzie. *Human-Computer Interaction: An Empirical Research Perspective*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 2013.
- [51] I Scott MacKenzie, Tatu Kauppinen, and Miika Silfverberg. Accuracy Measures for Evaluating Computer Pointing Devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '01, pages 9–16, New York, NY, USA, 2001. ACM.

- [52] I. Scott MacKenzie, Abigail Sellen, and William A. S. Buxton. A comparison of input devices in element pointing and dragging tasks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '91, pages 161–166, New York, NY, USA, 1991. ACM.
- [53] I. Scott MacKenzie and Colin Ware. Lag as a determinant of human performance in interactive systems. In *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems*, CHI '93, pages 488–493, New York, NY, USA, 1993. ACM.
- [54] Justin Matejka, Tovi Grossman, and George Fitzmaurice. Swift: Reducing the effects of latency in online video scrubbing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 637–646, New York, NY, USA, 2012. ACM.
- [55] Daniel M. Merfeld. Signal detection theory and vestibular thresholds: I. basic theory and practical considerations. *Experimental Brain Research*, 210(3):389–405, May 2011.
- [56] David E Meyer, Richard A Abrams, Sylvan Kornblum, Charles E Wright, and JE Keith Smith. Optimality in human motor performance: ideal control of rapid aimed movements. *Psychological review*, 95(3):340, 1988.
- [57] newzoo: Global games market report 2018. <https://newzoo.com/insights/trend-reports/newzoo-global-games-market-report-2018-light-version/>, June 2018. Accessed: 2019-03-07.
- [58] Albert Ng, Julian Lepinski, Daniel Wigdor, Steven Sanders, and Paul Dietz. Designing for Low-latency Direct-touch Input. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*, UIST '12, pages 453–464, New York, NY, USA, 2012. ACM.
- [59] James Nichols and Mark Claypool. The effects of latency on online madden nfl football. In *Proceedings of the 14th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, NOSSDAV '04, pages 146–151, New York, NY, USA, 2004. ACM.
- [60] Youtube: osu! liveplay meiko nakamura - dispel [insane]. <https://www.youtube.com/watch?v=goWG6us88Ag>, July 2013. Accessed: 2019-03-07.
- [61] Lothar Pantel and Lars C. Wolf. On the impact of delay on real-time multiplayer games. In *Proceedings of the 12th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, NOSSDAV '02, pages 23–29, New York, NY, USA, 2002. ACM.
- [62] Andriy Pavlovych and Carl Gutwin. Assessing target acquisition and tracking performance for complex moving targets in the presence of latency and jitter. In *Proceedings of Graphics Interface 2012*, GI '12, pages 109–116, Toronto, Ont., Canada, Canada, 2012. Canadian Information Processing Society.
- [63] Andriy Pavlovych and Wolfgang Stuerzlinger. Target Following Performance in the Presence of Latency, Jitter, and Signal Dropouts. In *Proceedings of Graphics Interface 2011*, GI '11, pages 33–40, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2011. Canadian Human-Computer Communications Society.
- [64] Michael Pettitt, Gary Burnett, and Alan Stevens. An extended keystroke level model (klm) for predicting the visual demand of in-vehicle information systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '07, pages 1515–1524, New York, NY, USA, 2007. ACM.

- [65] Peter Quax, Patrick Monsieurs, Wim Lamotte, Danny De Vleeschauwer, and Natalie Degrande. 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, pages 152–156, New York, NY, USA, 2004. ACM.
- [66] Kjetil Raaen and Andreas Petlund. How Much Delay is There Really in Current Games? In *Proceedings of the 6th ACM Multimedia Systems Conference*, MMSys '15, pages 89–92, New York, NY, USA, 2015. ACM.
- [67] Adi Robertson. Inside the desperate fight to keep old tv's alive - the verge. <https://www.theverge.com/2018/2/6/16973914/tvs-crt-restoration-led-gaming-vintage>, February 2018. Accessed: 2019-01-29.
- [68] Richard M Ryan and Edward L Deci. Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being. *American psychologist*, 55(1):68, 2000.
- [69] Richard M Ryan, C Scott Rigby, and Andrew Przybylski. The Motivational Pull of Video Games: A Self-Determination Theory Approach. *Motivation and Emotion*, 30(4):344–360, dec 2006.
- [70] Cheryl Savery, Nicholas Graham, Carl Gutwin, and Michelle Brown. The effects of consistency maintenance methods on player experience and performance in networked games. In *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing*, CSCW '14, pages 1344–1355, New York, NY, USA, 2014. ACM.
- [71] Cheryl Savery, T. C. Nicholas Graham, and Carl Gutwin. The human factors of consistency maintenance in multiplayer computer games. In *Proceedings of the 16th ACM International Conference on Supporting Group Work*, GROUP '10, pages 187–196, New York, NY, USA, 2010. ACM.
- [72] C. Scott Rigby, Edward L. Deci, Brian C. Patrick, and Richard M. Ryan. Beyond the intrinsic-extrinsic dichotomy: Self-determination in motivation and learning. *Motivation and Emotion*, 16(3):165–185, Sep 1992.
- [73] Steven C Seow. Information theoretic models of hci: a comparison of the hick-hyman law and fitts' law. *Human-Computer Interaction*, 20(3):315–352, 2005.
- [74] Nathan Sheldon, Eric Girard, Seth Borg, Mark Claypool, and Emmanuel Agu. The Effect of Latency on User Performance in Warcraft III. In *Proceedings of the 2Nd Workshop on Network and System Support for Games*, NetGames '03, pages 3–14, New York, NY, USA, 2003. ACM.
- [75] Dane Stuckel and Carl Gutwin. The effects of local lag on tightly-coupled interaction in distributed groupware. In *Proceedings of the 2008 ACM Conference on Computer Supported Cooperative Work*, CSCW '08, pages 447–456, New York, NY, USA, 2008. ACM.
- [76] Wacom intuos: Creative pen tablet — wacom. <https://www.wacom.com/en-us/products/pen-tablets/wacom-intuos>. Accessed: 2018-09-15.
- [77] Colin Ware and Ravin Balakrishnan. Reaching for objects in vr displays: Lag and frame rate. *ACM Trans. Comput.-Hum. Interact.*, 1(4):331–356, December 1994.

- [78] Bei Yuan, Eelke Folmer, and Frederick C. Harris. Game accessibility: a survey. *Universal Access in the Information Society*, 10(1):81–100, Mar 2011.
- [79] Min Hong Yun, Songtao He, and Lin Zhong. Reducing Latency by Eliminating Synchrony. *Proceedings of the 26th International Conference on World Wide Web*, pages 331–340, 2017.
- [80] Yi Zhang, Ling Chen, and Gencai Chen. Globally synchronized dead-reckoning with local lag for continuous distributed multiplayer games. In *Proceedings of 5th ACM SIGCOMM Workshop on Network and System Support for Games*, NetGames '06, New York, NY, USA, 2006. ACM.

APPENDIX A

FORMS

A.1 Consent Form (Study 1)

Department of Computer Science,
University of Saskatchewan

Participant Consent Form

You are invited to participate in a research study entitled: **Measuring effects of input lag in high-speed videogames**

Researcher(s): Michael Long (Graduate Student), Department of Computer Science, University of Saskatchewan, (306) 380-3380, michael.long@usask.ca

Supervisor: Carl Gutwin (Faculty) Department of Computer Science, University of Saskatchewan, (306) 966-8646, gutwin@cs.usask.ca

Purpose(s) and Objective(s) of the Research:

- Learn effects of input lag on high-speed aiming tasks & create model between lag and performance in videogames

Procedures:

- Participants will come to the HCI lab (room Thorv 376) at the University of Saskatchewan and play a videogame where users click on white circles with various computer input devices. Gameplay data will automatically be logged, and the participant will be asked some demographic questions and their prior experiences with videogames.
- Please feel free to ask any questions regarding the procedures and goals of the study or your role.

Funded by: NONE

Potential Risks:

- There are no known or anticipated risks to you by participating in this research except for possible arm/hand/finger tiredness. Participants will be given periodic breaks to rest their hands as long as they like between tasks.
- Participants will be told the goal of the study after it is done and given instructions on how to obtain the results from the study in the future.

Potential Benefits:

- Game designers will be able to create more enjoyable games by knowing the relationship between performance and lag.

Compensation: 10\$ honorarium

Confidentiality: (see consent guidelines section 9)

- **Storage of Data:**
 - Data will be securely stored either in a locked filing cabinet in a locked office OR on a password protected computer. All data will be stored for 5 years with Carl Gutwin.
 - When the data no longer required, the data will be destroyed by shredding (paper) or electronic file deletion.

Right to Withdraw:

- Your participation is voluntary and you can answer only those questions that you are comfortable with. You may withdraw from the research project for any reason, at any time without explanation or penalty of any sort until August 20th, 2018 for which results may have been disseminated. After this date, it is possible that some form of research dissemination will have already occurred and it may not be possible to withdraw your data
- Should you wish to withdraw, any data collected from your will be immediately deleted/destroyed.

Follow up:

- To obtain results from the study, please email michael.long@usask.ca two months after the study has been completed.

Questions or Concerns:

- Contact the researcher(s) using the information at the top of page 1;
- This research project has been approved on ethical grounds by the University of Saskatchewan Research Ethics Board. Any questions regarding your rights as a participant may be addressed to that committee through the Research Ethics Office ethics.office@usask.ca (306) 966-2975. Out of town participants may call toll free (888) 966-2975.

Consent

Your signature below indicates that you have read and understand the description provided; I have had an opportunity to ask questions and my/our questions have been answered. I consent to participate in the research project. A copy of this Consent Form has been given to me for my records.

Name of Participant

Signature

Date

Researcher's Signature

Date

A copy of this consent will be left with you, and a copy will be taken by the researcher.

A.2 Consent Form (Study 2)

Department of Computer Science,
University of Saskatchewan

Participant Consent Form

You are invited to participate in a research study entitled: **Measuring effects of input lag in high-speed videogames**

Researcher(s): Michael Long (Graduate Student), Department of Computer Science, University of Saskatchewan, (306) 380-3380, michael.long@usask.ca

Supervisor: Carl Gutwin (Faculty) Department of Computer Science, University of Saskatchewan, (306) 966-8646, gutwin@cs.usask.ca

Purpose(s) and Objective(s) of the Research:

- Learn effects of input lag on high-speed aiming tasks & create model between lag and performance in videogames

Procedures:

- Participants will come to the HCI lab (room Thorv 376) at the University of Saskatchewan and play a videogame where users click on white circles with various computer input devices. Gameplay data will automatically be logged, and the participant will be asked some demographic questions and their prior experiences with videogames.
- Please feel free to ask any questions regarding the procedures and goals of the study or your role.

Funded by: NONE

Potential Risks:

- There are no known or anticipated risks to you by participating in this research except for possible arm/hand/finger tiredness. Participants will be given periodic breaks to rest their hands as long as they like between tasks.
- Participants will be told the goal of the study after it is done and given instructions on how to obtain the results from the study in the future.

Potential Benefits:

- Game designers will be able to create more enjoyable games by knowing the relationship between performance and lag.

Compensation: 10\$ honorarium

Confidentiality: (see consent guidelines section 9)

• **Storage of Data:**

- Data will be securely stored either in a locked filing cabinet in a locked office OR on a password protected computer. All data will be stored for 5 years with Carl Gutwin.
- When the data no longer required, the data will be destroyed by shredding (paper) or electronic file deletion.

Right to Withdraw:

- Your participation is voluntary and you can answer only those questions that you are comfortable with. You may withdraw from the research project for any reason, at any time without explanation or penalty of any sort until August 20th, 2018 for which results may have been disseminated. After this date, it is possible that some form of research dissemination will have already occurred and it may not be possible to withdraw your data
- Should you wish to withdraw, any data collected from your will be immediately deleted/destroyed.

Follow up:

- To obtain results from the study, please email michael.long@usask.ca two months after the study has been completed.

Questions or Concerns:

- Contact the researcher(s) using the information at the top of page 1;
- This research project has been approved on ethical grounds by the University of Saskatchewan Research Ethics Board. Any questions regarding your rights as a participant may be addressed to that committee through the Research Ethics Office ethics.office@usask.ca (306) 966-2975. Out of town participants may call toll free (888) 966-2975.

Consent

Your signature below indicates that you have read and understand the description provided; I have had an opportunity to ask questions and my/our questions have been answered. I consent to participate in the research project. A copy of this Consent Form has been given to me for my records.

Name of Participant

Signature

Date

Researcher's Signature

Date

A copy of this consent will be left with you, and a copy will be taken by the researcher.

A.3 Consent Form (Study 3)

Department of Computer Science,
University of Saskatchewan

Participant Consent Form

You are invited to participate in a research study entitled: **Measuring effects of input lag in high-speed videogames**

Researcher(s): Michael Long (Graduate Student), Department of Computer Science, University of Saskatchewan, (306) 380-3380, michael.long@usask.ca

Supervisor: Carl Gutwin (Faculty) Department of Computer Science, University of Saskatchewan, (306) 966-8646, gutwin@cs.usask.ca

Purpose(s) and Objective(s) of the Research:

- Learn effects of input lag on high-speed aiming tasks & create model between lag and performance in videogames

Procedures:

- Participants will come to the HCI lab (room Thorv 376) at the University of Saskatchewan and play a videogame like Space Invaders. Gameplay data will automatically be logged, and the participant will be asked some demographic questions and their prior experiences with videogames.
- Please feel free to ask any questions regarding the procedures and goals of the study or your role.

Funded by: NONE

Potential Risks:

- There are no known or anticipated risks to you by participating in this research except for possible arm/hand/finger tiredness. Participants will be given periodic breaks to rest their hands as long as they like between tasks.
- Participants will be told the goal of the study after it is done and given instructions on how to obtain the results from the study in the future.

Potential Benefits:

- Game designers will be able to create more enjoyable games by knowing the relationship between performance and lag.

Compensation: \$5 honorarium

Confidentiality: (see consent guidelines section 9)

- **Storage of Data:**
 - Data will be securely stored either in a locked filing cabinet in a locked office OR on a password protected computer. All data will be stored for 5 years with Carl Gutwin.
 - When the data no longer required, the data will be destroyed by shredding (paper) or electronic file deletion.

Right to Withdraw:

- Your participation is voluntary and you can answer only those questions that you are comfortable with. You may withdraw from the research project for any reason, at any time without explanation or penalty of any sort until July 10th, 2018 for which results may have been disseminated. After this date, it is possible that some form of research dissemination will have already occurred and it may not be possible to withdraw your data
- Should you wish to withdraw, any data collected from your will be immediately deleted/destroyed.

Follow up:

- To obtain results from the study, please email michael.long@usask.ca two months after the study has been completed.

Questions or Concerns:

- Contact the researcher(s) using the information at the top of page 1;
- This research project has been approved on ethical grounds by the University of Saskatchewan Research Ethics Board. Any questions regarding your rights as a participant may be addressed to that committee through the Research Ethics Office ethics.office@usask.ca (306) 966-2975. Out of town participants may call toll free (888) 966-2975.

Consent

Your signature below indicates that you have read and understand the description provided; I have had an opportunity to ask questions and my/our questions have been answered. I consent to participate in the research project. A copy of this Consent Form has been given to me for my records.

Name of Participant

Signature

Date

Researcher's Signature

Date

A copy of this consent will be left with you, and a copy will be taken by the researcher.

A.4 Study Script and Demographics Questions, Study 1

Study: Measuring effects of input lag in high-speed videogames

Script & Questions

michael.long@usask.ca

(306) 380-3380

Hi _____. Thanks for coming. In this study you will be playing a game for 45 minutes where you must click on the white circles as quickly as you can after they appear. Occasionally you will be asked a few questions, during which you may rest your hand for as long as you like. The game will get harder as you play, and both moving and clicking with the cursor may become delayed. After finishing several rounds, you will repeat the exact same gameplay, but with a different input device. The input devices are: mouse, touchscreen, drawing tablet and controller. This study is completely voluntary, and you will be able to withdraw at any time for any reason. If you'd like to see the results of this study, please email michael.long@usask.ca two months after this session. Please read this consent form before we proceed. Feel free to move the screen/input devices to make using them more comfortable

Questions asked before playing the game:

- How old are you?
- Gender?
- Occupation?
- Left or right-handed? Ambidextrous?
- What hand do you use a computer mouse with?
- On a scale of 1 to 5, with 1 none, being and 5 being expert, how do you rate your proficiency with...
 - a mouse – a touchscreen – a controller – a drawing tablet
- On an average week, how many hours do you spend using...
 - a mouse – a touchscreen – a controller – a drawing tablet
- On an average week, how many hours of videogames do you play?
- If they do play videogames...
 - What genres of games do you play?
 - Do you play any games involving controllers? (console games, PC games /w controllers)
 - Do you play any games requiring a quick reaction time, such as first-person shooters?
- Have you experienced lag when playing videogames? (Such as online gaming, using a slow computer, etc.)
- In what situations has lag been a problem?
- How do you deal with lag when you play with videogames?

Have them perform <https://www.humanbenchmark.com/tests/reactiontime> 10 times, recording the end/overall avg.

Boot up experiment program. Enter participant ID, and device ordering. Tell them to click on the white circles as quickly and accurately as they can. They can rest their hands as long as they like when answering questions, or when it says 'click on circle to start round'. The game will add delay as you go, and when a block of trials has been completed you will switch to a different input device. Click on circle in screen center to start.

Questions asked after each round of gameplay (all answered on a 5-point Likert scale)

- Since the last survey, I felt capable and effective when playing.
- Since the last survey, using the current input device was fun.
- Since the last survey, I put a lot of effort into those rounds.
- Since the last survey, how well I did was completely due to me.
- Since the last survey, I was frustrated by the task.
- Since the last survey, the cursor movement was responsive.

Questions asked after playing the game:

- Overall, which device did you prefer using?
- Overall, which device do you think you performed the best with?
- Overall, which device do you think was most resilient to the delay/lag you experienced?
- Did you have strategies for playing the game in general?
- Did you have any strategies for when the lag when moving your player become too high?

Record any noteworthy/strange behaviour, or if anything went wrong.

A.5 Study Script and Demographics Questions, Study 2

Study: Measuring effects of input lag in high-speed videogames

Script & Questions

michael.long@usask.ca

mjl899

(306) 380-3380

Opening Script:

Hi _____. Thanks for coming. In this study you will be playing around 25 minutes of a videogame like Space Invaders. You will be moving the mouse left and right, dodging red bullets WHILST shooting at the aliens at the top of the screen. Between each round of gameplay, you will be asked a few questions, and you may rest your hand for as long as you like. The game will get harder as you play, and moving your player ship may become delayed. This study is completely voluntary, and you will be able to withdraw at any time for any reason. If you'd like to see the results of this study, please email michael.long@usask.ca two months after this session. Please read this consent form before we proceed.

Questions asked before playing the game:

- How old are you?
- Gender?
- Occupation?
- Left or right-handed? Ambidextrous?
- What hand do you use a computer mouse with?
- On an average week, how many hours of videogames do you play?
- If they do play videogames...
 - What genres of games do you play?
 - Do you play any games requiring a quick reaction time, such as first-person shooters?
- Have you experienced lag when playing videogames? (Such as online gaming, using a slow computer, etc.)
- In what situations has lag been a problem?
- How do you deal with lag when you play with videogames?

Questions asked after each round of gameplay:

- I felt capable and effective when playing that last round. (Answered on a 5 point Likert scale)
- Playing that last round was fun. (Answered on a 5 point Likert scale)
- How well I did during that last round was completely due to me. (Answered on a 5 point Likert scale)

Questions asked after playing the game:

- Were you looking at the top half of the screen, or the bottom half most of the time?
- Were you focussing more on dodging red bullets, or shooting the invaders?
- Did you have strategies for playing the game in general?
- Did you have any strategies for when the lag when moving your player become too high?

A.6 Study Script and Demographics Questions, Study 3

Space Invaders Lag Study

Michael.long@usask.ca

(306) 380-3380

Setting up the environment

- Make sure the computer and monitor are both on and logged in
- Test if both keyboard and red MSI gaming mouse are working
- Make sure the .exe runs and is the correct resolution
- Have the reaction speed test open the browser

The participant arrives

Explain that you will be playing a game like Space Invaders for ~20 minutes.

Have them fill out the consent form, offer them a copy of the consent form (they probably won't take it)

Have them fill out the honorarium form

Pre-game questions

- How old are you?
- Gender?
- Occupation?
- Left or right-handed? Ambidextrous?
- What hand do you use a computer mouse with?
- On an average week, how many hours of videogames do you play?
 - If they do play videogames...
 - What genres of games do you play?
 - Do you play any games requiring a quick reaction time, such as first-person shooters?
- Have you experienced lag when playing videogames? (Such as online gaming, using a slow computer, etc.)
- In what situations has lag been a problem?
- How do you deal with lag when you play with videogames?

Have them do this reaction speed test twice: <https://www.humanbenchmark.com/tests/reactiontime>

Record both averages (smaller number on the screen)

The game

Launch the .exe

Enter the participants unique participant ID

Explain the game:

The player moves with the mouse. You want to avoid incoming RED bullets WHILST shooting the enemy invaders at the top of the screen. Left click or hold down the LEFT MOUSE button to shoot bullets. When you get hit by an enemy bullet you will be unable to shoot for half a second. The game will get harder as you play. There will be a practice round when the game gets harder. Data from practice rounds is not recorded. There will sometimes be a delay when moving your player model. The practice rounds have no delay. After each round the game will prompt the user to answer some questions, which are answered using the number keys. Feel free to take a break for as long as you like while answering these in-game survey questions.

Press spacebar when player is ready

Game will play like this: Practice round (0ms of lag), 0ms of lag, 100ms, 200ms, 300ms, 400ms, break. Repeat this cycle 4 times, with the enemy bullet speed increasing each time.

Post-game questions

- Were you looking at the top half of the screen, or the bottom half most of the time?
- Were you focussing more on dodging red bullets, or shooting the invaders?
- Did you have strategies for playing the game in general?
- Did you have any strategies for when the lag when moving your player become too high?

Cleanup

- Note down any unusual findings/remarks from participant
- Did anything go wrong while running this participant? Note anything down
- Copy and paste the Invaders.csv file found in the MultiInputPong_Data folder onto the provided USB stick just in case
- Close and start the game back up (just in case)


If something unexpected happens during the study, feel free to come find me at my desk

A.7 Example In-Game Questionnaire Screen, Study 2

Read the question carefully, but try to answer as quickly as possible.

Playing that last round was fun.

(press 1,2,3,4,5 on your keyboard)



1 2 3 4 5

Strongly Disagree Disagree Neutral Agree Strongly Agree