

Configuration Tool and Experimental Platform for Pointing Devices

Jiawei Jin

Helsinki March 21, 2014

UNIVERSITY OF HELSINKI
Department of Computer Science

Tiedekunta — Fakultet — Faculty		Laitos — Institution — Department	
Faculty of Science		Department of Computer Science	
Tekijä — Författare — Author			
Jiawei Jin			
Työn nimi — Arbetets titel — Title			
Configuration Tool and Experimental Platform for Pointing Devices			
Oppiaine — Läroämne — Subject			
Computer Science			
Työn laji — Arbetets art — Level		Aika — Datum — Month and year	
Master's Thesis		March 21, 2014	
		Sivumäärä — Sidoantal — Number of pages	
		45 pages	
Tiivistelmä — Referat — Abstract			
<p>In user studies of human-computer interaction, experiments on new devices and techniques are often made on experiment software, which is developed separately for each device and technique. A systematic experimental platform, capable of running experiments on a number of technologies, would facilitate the design and implementation of such experiments. To do this, a configurable framework was created to allow relative pointing and absolute pointing input to be enhanced with adaptive pointing and smoothed pointing techniques. This thesis discusses both the internals of the framework as well as how a platform is developed based on the framework. Additionally, two calibration modules were designed to transform the relative pointing input to absolute pointing and obtain the necessary parameters which will be applied in smoothed pointing. As a part of the deployment, the experiment module was made to provide a platform which allowed the enhanced pointing experience to be evaluated and generated proper output according to the results of the experiment task.</p> <p>One key achievement presented in this thesis is that the relative pointing devices are integrable with adaptive pointing and smoothed pointing which support for absolute pointing devices in general. Another key result presented in this thesis is that the configurable framework based experimental platform provides proper functions which meet the demands of professional pointing evaluation.</p> <p>ACM Computing Classification System (CCS): I.4.1 [Digitization and Image Capture]: Camera calibration, I.4.3 [Enhancement]: Smoothing, I.4.8 [Scene Analysis]: Tracking</p>			
Avainsanat — Nyckelord — Keywords			
experimental platform, calibration, pointing device, adaptive pointing, smoothed pointing, Fitts's Law			
Säilytyspaikka — Förvaringsställe — Where deposited			
Muita tietoja — övriga uppgifter — Additional information			

Dedication

This dissertation is dedicated to my family and girlfriend. A special feeling of gratitude to my loving parents, Zhong Jin and Yuhong Ruan. Without their words of push and encouragement tenacity ring in my ears, my academic goals would be much more difficult to achieve.

I also dedicate this dissertation to my girlfriend Yanshan Wang because she always believed in my work and was there at the best and worst moments. This thesis is as much yours as mine.

Acknowledgments

I am grateful to a number of people who have helped make this research possible. I would first like to appreciate Prof. Giulio Jacucci for presenting me with the opportunity to research in this fantastic field and supervising this thesis. Secondly, I would like to thank my co-supervisor Dr. Antti Jylhä for supporting and advising me regarding the problem description as well as his review of parts of my thesis.

Finally, I would like to thank Imtiaj Ahmed for his endless ideas and corrections. Thank you very much for using your precious time at any time of the day.

List of Figures

1	3DConnexion SpaceBall 5000 Trackball. [3Dc14]	5
2	Leap Motion. [LM14]	7
3	Player Interacts with Nintendo Wii. [Lim13]	8
4	CyberGlove III. [CS10]	9
5	Sequence Diagram of Experiment Platform.	11
6	Linear Function of Calibration Formula.	13
7	Experiment with 16 Objects.	23
8	User Activity.	25
9	Layout of User Interface.	29
10	Manually Parameter Setting.	30
11	Experiment Configuration.	31
12	Crosshair for Calibration.	32
13	Layout of Experiment.	33
14	Output of Experiment.	34

Contents

Dedication	ii
Acknowledgments	iii
List of Figures	iv
1 Introduction	1
2 Pointing Interactions and Devices	3
2.1 Relative Pointing	3
2.2 Absolute Pointing	4
3 System Principles	10
3.1 Platform Architecture	10
3.2 Calibration Algorithm	12
3.3 Adaptive Pointing	14
3.4 Smoothed Pointing	17
3.5 Fitts's Law	21
3.6 Experiment Principle and Algorithm	22
4 Implementation	25
4.1 Design	25
4.2 Hardware	27
4.3 Software and Libraries	28
4.4 User Interface	29
4.5 Calibration	30
4.6 Experiment	32
5 Evaluation	35
5.1 Apparatus	35
5.2 Procedure	36
5.3 Result	37
5.4 Discussion	38
6 Conclusions and Further Work	40

6.1	Conclusions	40
6.2	Future Research and Extensions	41
	References	42

1 Introduction

Since the earliest graphical user interface (GUI) was introduced, the mouse has been the most used pointing device. The mouse can be used to move through the user interface and select or drag an item in an effective and accurate way. However, the mouse is no longer the only pointing device that can be used. More and more pointing devices with different technologies are being invented. For instance, the touch screen is widely used on smart phones, tablets, and laptops or even desktops recently and the mouse is not the only way that can be used to interact with GUI. The mouse becomes a limited pointing device for some applications since it can only be controlled by one hand and the tasks that the buttons can achieve are very limited. People want to control GUI by their fingers, arms, bodies and eyes. In order to meet people's needs, many camera-based pointing devices were designed, such as Leap Motion, Kinect and Tobii X120 [PAC13][FPT12]. Using these pointing devices in a natural way is a problem. Some applications like painting require that human's behavior should be expressed in visually follow the right path while other applications pay attention on the stableness of the output. In order to examine the efficiency, accuracy and correctness of different pointing devices with different pointing techniques, a systemic configurable framework is needed.

During the last twenty years, most pointing interface experiments were based on International Organization for Standardization (ISO) 9241 Ergonomics of human-system interaction, Part 400 Principles and requirements for physical input devices [ISO10] compliant circular Fitt's Law. The primary motivation of the standard is to improve the user experience of computer pointing devices to cater to user's physiological capabilities and limitations. Meanwhile, the standard introduces uniform testing procedures and guidelines to evaluate the performance of pointing devices produced by different manufacturers [ISO10].

On the other hand, new pointing techniques were introduced to improve the performance of pointing experience. Konig presented a precision enhancing technique to reduce the error rate while users tried to point at a tiny object, namely Adaptive

Pointing [KGDR09]. Adaptive pointing reduced the velocity of the pointer cursor when the cursor was moving in low speed and a high precision was needed. However, the nature of adaptive pointing violated the assumption of absolute pointing that displayed the pointer cursor in the right place where a user mentally wanted to point at. Two years later, Gallo and Minutolo improved Adaptive Pointing so that the pointer could follow the exact path that a user drew when ensuring the stableness of the pointer, this new technique was called Smoothed Pointing [GM12]. Nevertheless, neither Adaptive Pointing nor Smoothed Pointing was a part of ISO 9241-400.

The aim of this thesis is to build an experimental platform which is able to generate experimental data for three pointing devices (Leap Motion, Kinect and Miramatrix) based on a proposed configurable framework. The framework allows any relative pointing or absolute pointing input to be enhanced with adaptive pointing or smoothed pointing technique either through a calibration task or manually configurations. The enhanced pointing data will be tested by an experiment task in the experimental platform. At the same time, a model of human movement called Fitts's Law is used as the principle of the experiment task [AZ97].

This thesis is organized in six sections. Section 2 provides the background of the research work. It introduces several absolute pointing devices and relative pointing devices. Section 3 introduces the conceptual design idea of the platform architecture. The algorithms of Adaptive Pointing and Smoothed Pointing as well as the principle behind the experiment task are discussed. Section 4 shows how the platform is implemented. The hardware and software of the implementation are introduced. The calibration methods, the appearance of the application and the experiment task are also included. In section 5, the platform is evaluated by gathering feedback from several members in the research group and the results of the feedback are summarized. Section 6 presents the concluding remarks as well as a brief analysis of the results. At the end, future work is discussed.

2 Pointing Interactions and Devices

In order to use any kind of computers, as the end user of computer program, human must communicate with a computer by sending messages to it or by receiving feedback from it. These kind of exchanges are made by human through either an input device or an output device depending on whether the information is going to a computer or comes from a computer. Input devices are used to specify actions and send information to computers and the output devices allow computers to show the status or results of corresponding information. According to Preece [PRS⁺94], input devices can be divided into two categories: keyboards and pointing devices.

Keyboards are the most popular input devices used to enter information into computers. The number of possible commands is limited since the number of keyboard keys which are used to produce letters, numbers or signs is finite. Pointing devices are used to specify a state or a position in one, two or three dimensions. Computers display the pointer continually according to the user action caught by the pointing devices. Certainly, the number of possible actions that a pointing device can recognize is also limited.

Currently, lots of applications use both a keyboard and a pointing device together. Normally, both of them are necessary, but some commands can be performed by either a keyboard or a pointing device. For instance, icons on the desktop can be selected by a keyboard or a mouse. Which device will be chosen depends on users' preference.

2.1 Relative Pointing

Using relative pointing devices is a good choice when the available surface or space is limited. Relative pointing devices let users move the cursor from a start position to an ending position. Instead of mapping between a device and the coordinate on the screen directly, relative pointing devices catch the coordinate in the motor area and map into a corresponding coordinate on the visual area. The speed of moving

a cursor can be adjusted by changing the value of the Control-Display Gain (CD-gain). CD-gain is a unit free coefficient that maps the movement of the pointing device to the movement in the display space [CVPC07].

3DConnexion SpaceBall 5000 (Figure 1) is a representative example, it is a relative pointing device with 12 buttons and one trackball that allows the user to control the rotation in applications and move a pointer cursor in 3 dimensions [DRBS90]. The movement is performed by scrolling the rubber ball in desired direction while the rotation is produced by rotating the ball horizontally. The SpaceBall can handle a large movement in a very precise way since it can increase the CD-gain automatically when fast speed scrolling is detected. Similar to most relative pointing devices, the surface that the SpaceBall needs is relatively smaller than an absolute device [DRBS90] needs. The SpaceBall is usually used as a succedaneum of traditional keyboard and mouse when users want to perform a navigation task in a 3D application but only need several functional keys.

2.2 Absolute Pointing

Unlike relative pointing devices, absolute pointing devices require a larger surface or space for reasonable movement. A pointer cursor will be shown at the exact place that the user tries to point at. Laser pens are such devices; no matter how a user moves the laser pen, the dot will always appear straight along the laser that comes from the nib. However, absolute pointing devices like laser pen do not support any enhanced pointing technology. Instead of using physically absolute pointing devices, people usually perform a 3D tracking to obtain absolute pointing by a relative device. Although relative pointing devices do not support absolute pointing by default, 3D tracking systems can provide an absolute measurement with a relative input.

3D tracking uses sensors to monitor the key points or joints of human's finger, hand, body or eyes and calculates the corresponding coordinate according to the collected information. There are mainly three technologies that can perform a 3D tracking used in absolute measurement field, which are Video Tracking, Infrared



Figure 1: 3DConnexion SpaceBall 5000 Trackball. [3Dc14]

(IR) Tracking and Mechanical Tracking.

Video tracking is a low-cost tracking technology since the required devices are limited to cameras. Video tracking uses one camera or multiple cameras to locate a moving object over time. The algorithm that performs video tracking analyzes continuous video frames and output the movement of target objects between two frames [SxQIH07]. Video tracking is highly active, because the transmission speed from the sensor to the object is the speed of light and the frames are only updated for thirty to sixty times in a second [SxQIH07]. The update time duration between two frames

allows the pointing devices to have time to transmit the image to a computer and the computer have time to generate the output following a specific algorithm. Thus, the more responsive the sensors and the computer are, the less delay the user will recognize. Besides, video tracking does not need further equipment attached, the whole tracking process is handled by the tracking algorithms regardless of what object is being tracked. Nevertheless, the quality of caught image affects the accuracy of output. It depends on the environment lighting conditions and the distance between cameras and objects.

Leap Motion (Figure 2) is a device that uses video tracking. It is a camera-based pointing device that requires no hand contact or touching. The hand, finger or tool motions are recognized by the sensor as input. The position of fingertip, the direction of finger and the distance between the palm and fingertip, all these kind of information related to users' hand is transformed into numeric value and delivered to an application. Meanwhile, the application analyzes the collected information and displays the output according to the algorithm it performs.

IR Tracking is a lineal successor of video tracking. Since IR energy is emitted from all things that have a temperature greater than absolute zero, the sensor used for IR tracking does not have the problem with lighting condition, everything exists on the earth can be detected by the IR sensor [WBA⁺13]. However, having this advantage does not mean that IR tracking is better than standard video tracking. Compare to the video tracking device, the relative higher capitalized cost of IR sensor is still the main reason why most domestic device manufacturers prefer to produce standard video tracking devices rather than IR tracking devices. On the other hand, some detection problems occur when an IR sensor is trying to detect the position of objects and other IR sources are around at the same time, those IR sources can be candles or incandescent light bulbs [Cas06].

Nintendo Wii remote controller and sensor bar provide a platform for players so that at most four players can be tracked by the IR sensors connected to the game console regardless of television's type or size. Figure 3 shows that a player is interacting

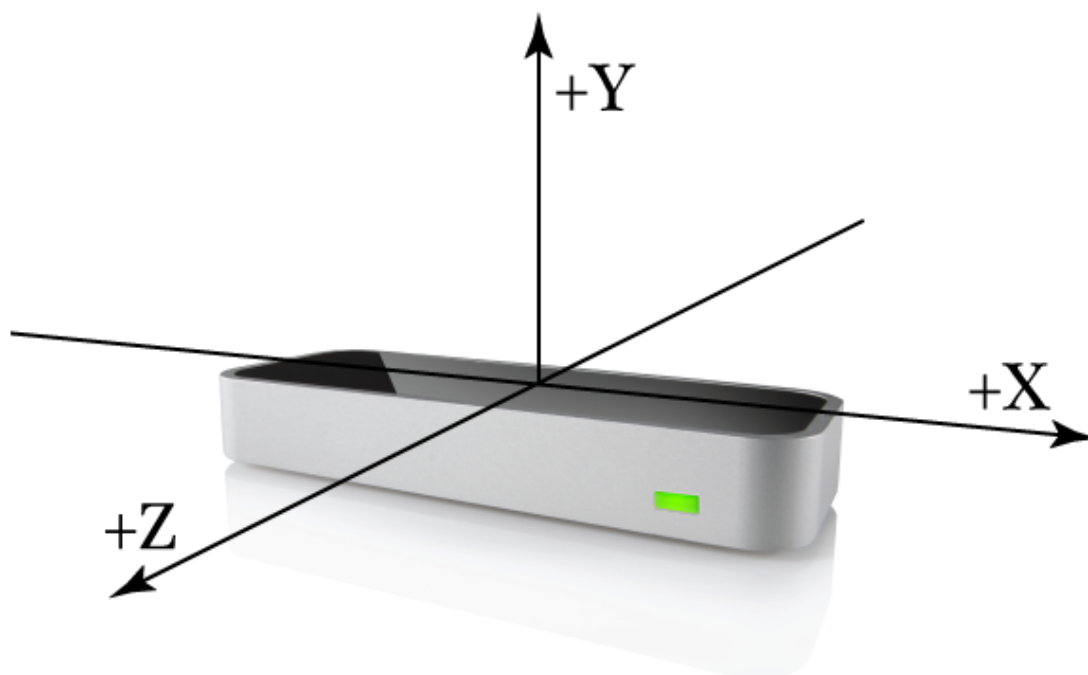


Figure 2: Leap Motion. [LM14]

with Nintendo Wii. The player holds a Wii remote controller and try to aim at the sensor bar placed upon a television. The Wii remote controller senses light from the sensor bar and outputs the coordinate and size of the four brightest IR points it recognizes [Ett11].

Mechanical Tracking is mainly used for motion capture which tracks body joint angles. In order to use mechanical tracking, some sensors are usually attached to the body, this is the reason why mechanical tracking is always referred to as Exoskeleton motion tracking [PB02]. Haptic feedback is useful in a mechanical tracking system when the system needs to guide users' behavior. Unlike video tracking, mechanical tracking system does not need to care about how bad the environment is, since all motions are captured by sensors and the sensors communicate with the mechanical tracking system through wired or wireless connections. Due to this advantage, mechanical tracking is very accurate and the caught motions can be updated in a high frequency. The disadvantage of mechanical tracking is obviously the weight of

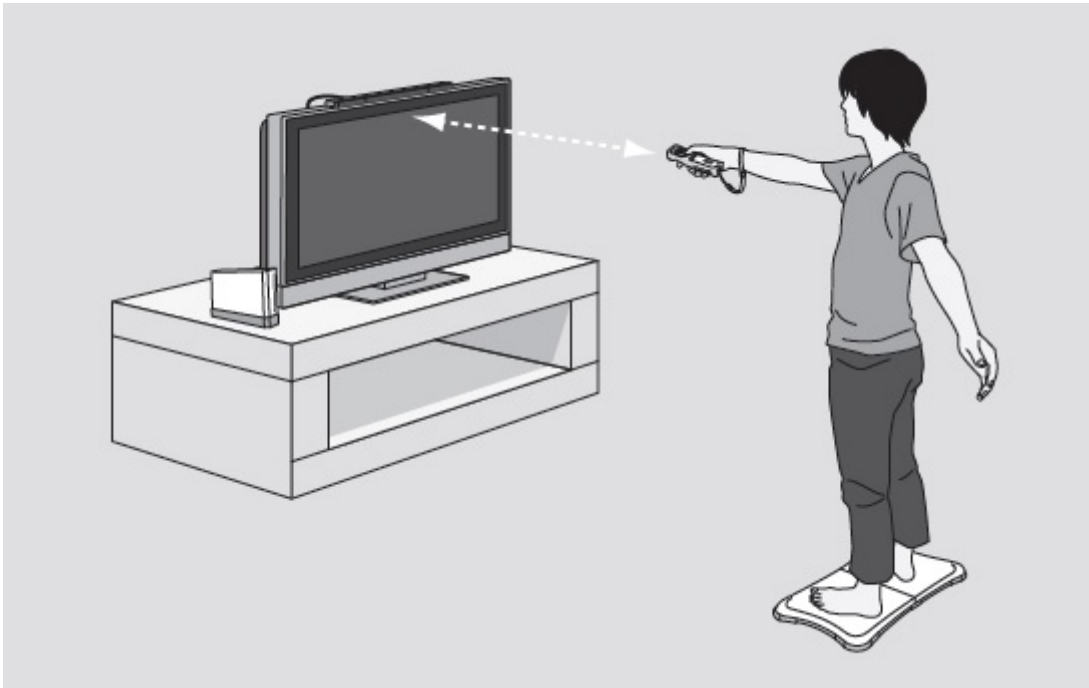


Figure 3: Player Interacts with Nintendo Wii. [Lim13]

sensors. Some system requires user to be equipped by rigid structures of straight, jointed metal or plastic rod connected together. Those equipments usually weigh 3 kilograms (approximately 6.6 pounds) or even more than 3 kilograms, it is a hard pressure for users if they need to do a series of continuous actions [PB02].

As a well-known mechanical tracking system. CyberGlove III (Figure 4) uses fifteen flexion sensors, four abduction sensors and a palm-arch sensor to measure the hand and finger positions with less than 1 degrees resolution in a minimum 90 records/sec sensor data rate [KHW95]. It is usually used as a whole hand input device when hand and arm gestures recognition are important to users [KHW95].



Figure 4: CyberGlove III. [CS10]

3 System Principles

The goal of this thesis is to build a framework which provides an experimental platform for different pointing devices in order to test their performances with different pointing techniques. Two enhanced pointing techniques, adaptive pointing and smoothed pointing are embedded in the framework. The calibration techniques that the platform uses will be discussed. The concept behind the Fitts's Law as well as the experiment relies on it will also be explained.

3.1 Platform Architecture

Altogether eight modules operate within the framework. These modules include a user interface module, two calibration modules, an experiment module, two pointing techniques modules and one gesture module. The sequence diagram (figure 5) shows the relationships among these modules and how they communicate with each other.

As the interactive interface of the system, the user interface is responsible for providing users options to change the parameters of the experiment and displaying the latest values of those parameters.

Before the input data can be processed, each pointing device should offer absolute pointing measurement by default. Otherwise, a calibration is needed for transforming relative pointing measurement data into absolute pointing measurement data. Since most relative pointing devices do not have the global view of their own geographical position, it is impossible for them to determine the distance between the display and themselves. Thus, relative pointing devices need to calculate the offset and calibrate the pointer cursor in the display space. The algorithm of the calibration for relative pointing devices is discussed in section 3.2. Furthermore, before the input coordinates can be enhanced by adaptive pointing and smoothed pointing, another calibration must be applied on all pointing devices so that the needed information for smoothed pointing can be collected by the platform.

The experiment module can be considered as a core of the whole system. It in-

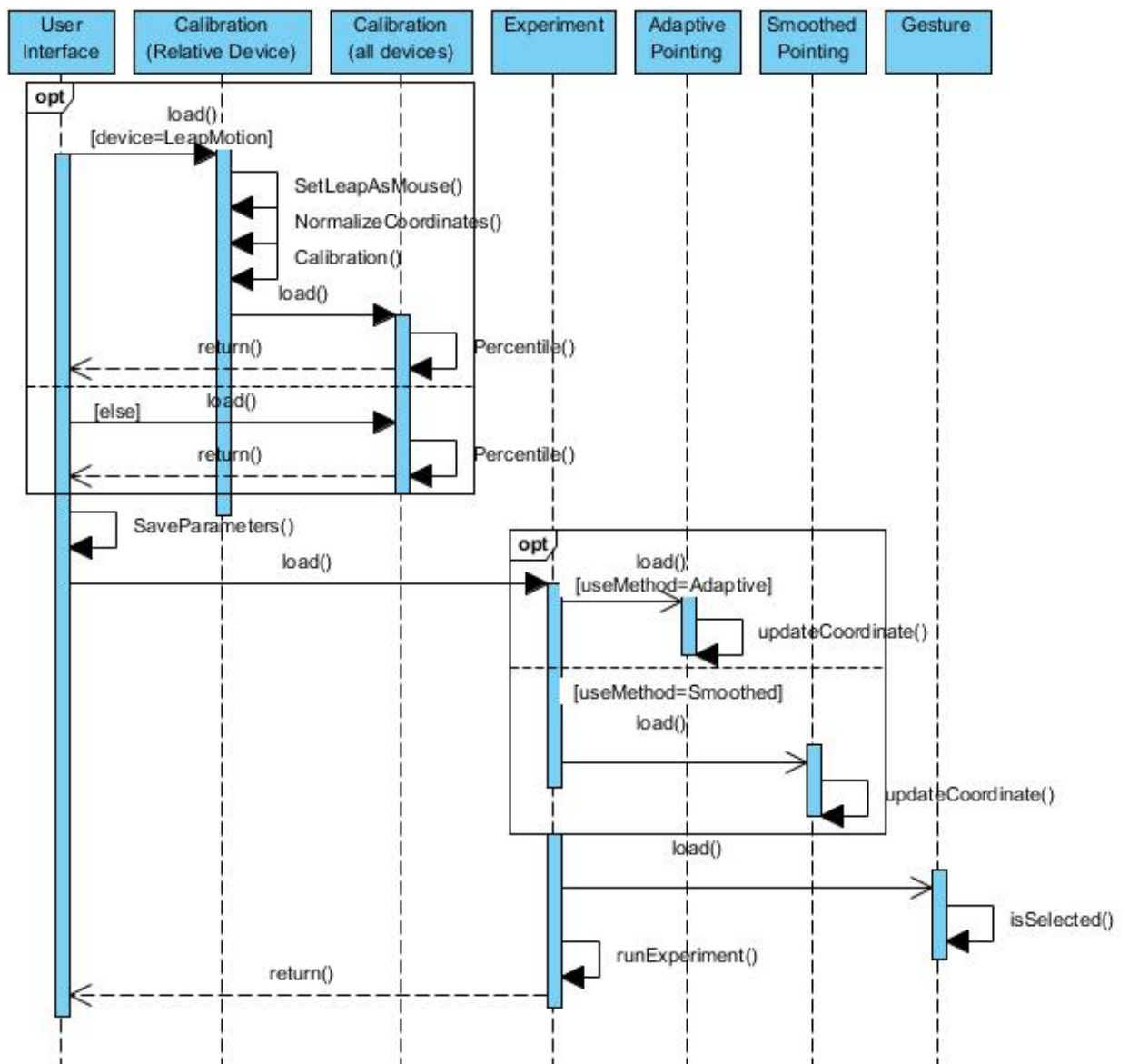


Figure 5: Sequence Diagram of Experiment Platform.

stantiates adaptive pointing module, absolute pointing module and gesture modules and forms experiment tasks for the framework based on Fitts's Law according to the information obtained from the user and the input pointing device. Whenever an experiment is performed, the experiment module starts to record all necessary data and stores these values into a log file. Once a series of tasks are done by the user, the experimental platform will return to the user interface and be ready for the next round.

In order to have a global control power of the whole system, the platform maintains a script which stores all the variables needed by all the modules. The parameters modified in the user interface, the transformed coordinate data and the variables used during the experiment are all stored in this script.

3.2 Calibration Algorithm

In order to be aware of where a user is trying to point at on a display screen, we must first "teach" the system what the user's actions look like when the user is pointing at known locations on the display screen.

A two-point calibration is used as the main calibration technique for the relative pointing devices. The two-point calibration is more accurate than a one-point calibration. The one-point calibration can only calibrate one point from the motor space to the display space, but any other points will skew since this calibration technique has no awareness about how big the offset is. The farther a point is located away from the calibrated point in the motor space, the more it will skew from the actual point that it should be displayed. By using the two-point calibration, normalized size of the screen height and width should be calculated so that the coordinates in the motor space can be mapped into the display space accurately. The points on the lower left and upper right of the display screen are measured and the calibration follows the equation below:

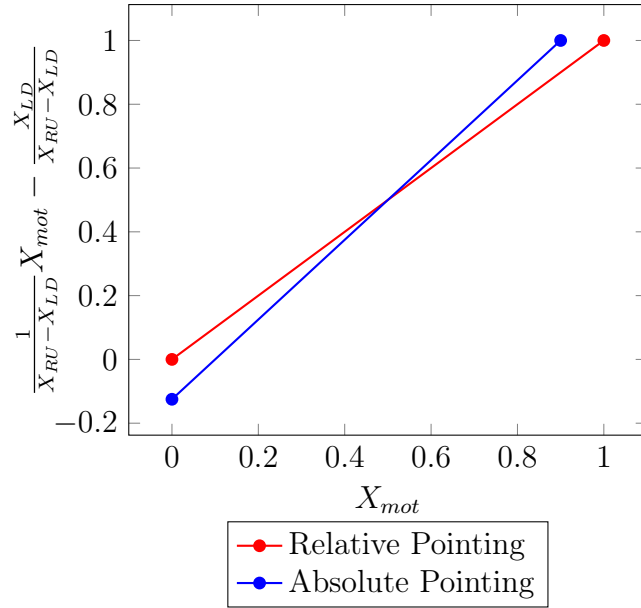


Figure 6: Linear Function of Calibration Formula.

$$X_{disp} = \frac{(X_{mot} - X_{LD})}{X_{RU} - X_{LD}}$$

X_{RU} and X_{LD} represent the X axis coordinates of the right upper point and left lower point respectively while X_{mot} represents the X axis coordinate of the current pointer cursor in the motor space and X_{disp} represents the X axis coordinate of the current pointer cursor in the display space. The codomain of X_{mot} is from 0 to 1. The value of X_{mot} can neither lower than 0 nor greater than 1. Figure 6 shows the equivalent graph of the linear function.

The red line and blue represent the one-to-one relationships between the X axis coordinates in the motor space and the display space. The red line shows the relative pointing when the value of X_{RU} equals to 1 and the value of X_{LD} equals to 0, the X coordinate in the display space remains the same as it in the motor space. Nevertheless, after a relative pointing device has been calibrated by two-point calibration, the input data will be transformed into the absolute pointing measurement data. For instance, the blue line shows the relationship of X_{RU} and X_{LD} when the value of X_{RU} reduces to 0.9 and the value of X_{LD} increases to 0.1.

At this moment, the X axis coordinates in the motor space and the display space become different, absolute pointing is now performed. The same algorithm also applies to the Y axis coordinates in the motor space and the display space.

Although the two-point calibration can be used for eye-tracking pointing input, the system can no longer ensure the correctness of transforming relative pointing data into absolute pointing data since the allowed movement space of human's pupil is relatively too small compare with the ordinary screen size. An one millimeter movement of human's pupil can be enlarged to 70 millimeters when the movement is mapped to display space with absolute pointing measurement [CVC12]. Thus, a more precise calibration is need for eye-tracking. According to Cerrolaza, a standard calibration set of eye-tracking usually consists of 9 points distributed as a 3x3 grid and the polynomial can be defined as [CVC12]:

$$\begin{cases} X_{disp} = a_0 + a_1X_{mot} + a_2Y_{mot} + a_3X_{mot}Y_{mot} + a_4X_{mot}^2 + a_5Y_{mot}^2 \\ Y_{disp} = b_0 + b_1X_{mot} + b_2Y_{mot} + b_3X_{mot}Y_{mot} + b_4X_{mot}^2 + b_5Y_{mot}^2 \end{cases}$$

where the coefficients $a_0...a_5$ and $b_0...b_5$ are the unknown values and can be computed using least squares.

The platform supports both relative pointing input and absolute pointing input. However, relative pointing devices need an extra calibration to obtain absolute pointing measurement since both adaptive pointing and smoothed pointing are based on absolute pointing [KGDR09][GM12]. As a part of smoothed pointing technique, the extra calibration for absolute pointing devices will be discussed in section 3.4.

3.3 Adaptive Pointing

Absolute pointing provides a position-to-position mapping which offers the user a more natural pointing experience and convenient hand-eye coordination compared with relative pointing. However, absolute pointing suffers from the precision problem caused by the distance between the pointing device and the display screen. In order

to overcome the drawback of absolute pointing, König introduced a velocity-oriented approach, namely adaptive pointing [KGDR09].

The basic idea of adaptive pointing is to improve the performance of absolute pointing measurement while ensuring that users mentally realize the absolute pointing operation is performed [KGDR09]. Users desire a one-to-one mapping pointing experience between the motor space and the display space. Adaptive pointing provides a natural absolute pointing behavior and enhances the precision of pointing by adjusting the CD-gain of a pointer cursor whenever a higher precision or an absolute pointing measurement is needed [KGDR09]. The decision of the CD-gain adjustment depends on the present velocity of the pointing movement and the offset between the motor space and the display space. The range of adjustable CD-gain is limited between g_{min} and g_{max} while v_{min} and v_{max} define the velocity bounds when the algorithm is notified to adjust the CD-gain. The following equations are introduced by König [KGDR09], only the X axis coordinate $x_{disp}(t)$ in the display space will be discussed. The same algorithm applies for the Y axis coordinate $y_{disp}(t)$ in the display space.

$$\hat{v}_x(t) = \begin{cases} 1 & \text{if } v_x(t) > v_{max} \\ 0 & \text{if } v_x(t) < v_{min} \\ \frac{v_x(t) - v_{min}}{v_{max} - v_{min}} & \text{otherwise} \end{cases} \quad (1)$$

The equation 2 and 3 concern the offset of the point coordinates in the motor space and the display space in order to define the offset bounds in normalized values using the same method as in equation 1:

$$d_x(t) = x_{mot}(t) - x_{disp}(t - 1) \quad (2)$$

$$\hat{d}_x(t) = \begin{cases} 1 & \text{if } |d_x(t)| > d_{max} \\ 0 & \text{if } |d_x(t)| < d_{min} \\ \frac{d_x(t) - d_{min}}{d_{max} - d_{min}} & \text{otherwise} \end{cases} \quad (3)$$

where d_{min} and d_{max} are the offset bounds and $d_x(t)$ is the current offset in X axis coordinate in normalized values. Meanwhile, equation 4 compares these two factors and chooses the greater one to be used in the next operation.

$$m_x(t) = \max(\hat{v}_x(t), \hat{d}_x(t)) \quad (4)$$

In order to avoid unexpected switching from a relative mapping to a absolute mapping or conversely, equation 5 represents a modulated sine wave to smooth the switching. This feature makes the adjustment still more natural.

$$g_x(t) = g_{min} + \frac{1}{2}[\sin(m_x(t) \cdot \pi - \frac{\pi}{2}) + 1](g_{max} - g_{min}) \quad (5)$$

Then the last movement in the motor space is computed as in equation 6.

$$s_x(t) = x_{mot}(t) - x_{mot}(t - 1) \quad (6)$$

Meanwhile, the most recent CD-gain will be decided in equation 7 as follow:

$$\hat{g}_x(t) = \begin{cases} 1 - (g_x(t) - 1) & \text{if } g_x(t) > 1 \text{ AND } d_x(t) > 0 \text{ AND } s_x(t) < 0 \\ 1 - (g_x(t) - 1) & \text{if } g_x(t) > 1 \text{ AND } d_x(t) < 0 \text{ AND } s_x(t) > 0 \\ g_x(t) & \text{otherwise} \end{cases} \quad (7)$$

Finally, equation 8 will apply the current CD-gain to the movement $s_x(t)$ and adds the value of the last coordinate $x_{disp}(t - 1)$ so that the current X axis coordinate in the display space can be calculated.

$$x_{disp}(t) = x_{disp}(t - 1) + \hat{g}_x(t) \cdot s_x(t) \quad (8)$$

According to the algorithm, if a user reduces the velocity of a pointer cursor less than v_{max} insistently, the CD-gain will be decreased smoothly as well. Until the defined minimum velocity v_{min} is reached, the CD-gain will be fixed at once even the actual velocity is lower than v_{min} . Similarly, if the user increases the velocity insistently, the CD-gain will be increased until a defined maximum CD-gain g_{max} is reached. However, the value of g_{max} is defined greater than 1 in general [KGDR09]. It means that the CD-gain will be always greater than the one used in the absolute pointing measurement even the user moves the pointer cursor in high speed all the time. The character of adaptive pointing violates the nature of absolute pointing. Although adaptive pointing improves the accuracy of pointing experience when high precision is needed, it can not guarantee that the user can perform an unadulterated absolute pointing measurement at other times.

3.4 Smoothed Pointing

Smoothed pointing was introduced by Gallo [GM12] to overcome the weakness of adaptive pointing. It is very similar to adaptive pointing which adjusts CD-gain dynamically so that the precision can be improved in low-speed movement. However, unlike adaptive pointing, smoothed pointing also provides a pure absolute pointing measurement in high speed movement by applying offset recovery into the algorithm [GM12]. The offset recovery allows smoothed pointing to recover the offset engendered when the CD-gain has been reduced and then increases to 1.

On the other hand, smoothed pointing applies a calibration task for absolute pointing inputs. The task requires the user to concentrate on pointing at a specific point for a period of time. The distance between the pre-defined point and the pointer cursor as well as the velocity in each frame will be recorded. Suppose that the amount of the distance data or the velocity data collected in all frame is n and all the data

from k_1 to k_n are organized in ascending order. According to Delage [DM07], the equation 9 is concluded to calculate the value of 90 percentile d_{90} .

$$d_{90} = \begin{cases} \frac{k_{\frac{n}{90}} + k_{\frac{n}{90}+1}}{2} & \text{if } \frac{n}{90} \bmod 1 = 0 \\ \lceil k_{\frac{n}{90}} \rceil & \text{if } \frac{n}{90} \bmod 1 > 0.5 \\ \lfloor k_{\frac{n}{90}} \rfloor & \text{if } 0 < \frac{n}{90} \bmod 1 \leq 0.5 \end{cases} \quad (9)$$

For instance, a user is trying to pointing at the spot located in (400,500) and the data has been collected for 10 seconds. As a result, the calculated value of d_{90} is 45 pixels. It means that according to the user's habitual behavior, if the velocity of movement reduces to less than v_{max} and the distance between the current pointer cursor and the spot (400,500) is less than 45 pixels, then we can assume that the user is aiming at the spot (400,500). On the other hand, a minimum target size X_{meters} that a human with 6/6 vision is able to recognize from a distance D can be calculated by

$$X_{meters} = 2 \cdot D \cdot \tan\left(\frac{1}{120}^\circ\right) \quad (10)$$

where D is the distance between the user and the display screen. Since a display screen is used as the output, X_{meters} must be convert to X_{pixels} as

$$X_{pixels} = \frac{X_{meters} \cdot 100 \cdot k_{dpi}}{2.54} \quad (11)$$

where k_{dpi} is the dots per inch(dpi) of the screen. Meanwhile, the value of d_{max} can be easily found as hundredfold of X_{pixels} . As the value of d_{90} and X_{pixels} are known, the minimum allowed CD-gain g_{min} can be calculated by

$$g_{min} = \frac{X_{pixels}}{d_{90}} \quad (12)$$

The calculations of $d_x(t)$, $s_x(t)$ and $m_x(t)$ remain the same as in adaptive pointing.

However, instead of calculating the velocity $v(t)$ with X axis and Y axis separately, the velocity $v(t)$ now concerns the movement of the pointer cursor in both X and Y axes in each frame. The calculation is formulated as

$$v(t) = \frac{\sqrt{\Delta_x(t)^2 + \Delta_y(t)^2}}{T} \quad (13)$$

where $\Delta_x(t)^2$ and $\Delta_y(t)^2$ represent the movements in X and Y axes in one frame and T is the duration between two frames. In smoothed pointing, $\hat{v}_x(t)$ is no longer fixed to 0 or 1 no matter if the value of $v_x(t)$ is greater or less than v_{max} . According to Gallo, the value of v_{min} can be found from the results of the calibration task for smoothed pointing and the value of v_{max} is five times as much as v_{min} [GM12]. Meanwhile, the offset between the motor space and the display space $\hat{d}_x(t)$ is defined as

$$\hat{d}_x(t) = \frac{|d_x(t)|}{d_{max}} \quad (14)$$

A offset will be recovered when the CD-gain has been adjusted and the system tries to perform a pure absolute pointing measurement. Thus, the algorithm need to decide when $g_x(t)$ should be set to 1 and when $g_x(t)$ should be set a bit greater than 1. Instead of fixing the value of $g_{xmax}(t)$, Gallo presented a new formula to adjust the value of $g_{xmax}(t)$ [GM12]. The CD-gain $g_x(t)$ is formulated as

$$g_x(t) = \begin{cases} g_{min} + \frac{1}{2}[\sin(m_x(t) \cdot \pi - \frac{\pi}{2}) + 1](1 - g_{min}) & \text{if } v(t) \leq v_{max} \\ g_{xmax}(t) & \text{otherwise} \end{cases} \quad (15)$$

where

$$g_{xmax}(t) = \begin{cases} \frac{d_x(t)}{s_x(t)} + \frac{1}{\hat{v}(t)} \left(1 - \frac{d_x(t)}{s_x(t)}\right) & \text{if } d_x(t) \cdot s_x(t) > 0 \\ \frac{1}{\hat{v}(t)(1 + \hat{d}_x(t))} & \text{otherwise} \end{cases} \quad (16)$$

Eventually, as well as adaptive pointing, the mapped coordinate value is calculated by

$$x_{disp}(t) = x_{disp}(t - 1) + g_x(t) \cdot s_x(t) \quad (17)$$

Obviously, compare to adaptive pointing, the advantage of smoothed pointing is that it can recover the offset in a short period and switch to pure absolute pointing mode immediately. Certainly, the switching process is performed smoothly benefited from the flexible formula $g_{xmax}(t)$. Although smoothed pointing is able to adjust CD-gain dynamically and all the related parameters except the distance D can be measured automatically, smoothed pointing still has several weaknesses. According to equation 10, the algorithm assumes that all the users have a 6/6 vision by default. Therefore, the algorithm is not suitable for the users whose vision is lower or higher than 6/6. Furthermore, in spite of automatic configuration, this feature does not allow users to perform some tasks with specific requirement. For instance, it is impossible for a user to perform an experiment which allows high error rate in smoothed pointing mode if the error rate has been calculated according to the 90 percentile calibration task.

There is no standard that defines if smoothed pointing is better than adaptive pointing or conversely. Thus, both of them are embedded into the framework in order to meet the needs of the users with different purposes.

3.5 Fitts's Law

Fitts's Law is a model of human movement proposed by Paul Fitts mainly used in ergonomics and human-computer interaction (HCI) [AZ97]. Fitts's Law is used to model the act of pointing which measures the performance and correctness of pointing movement either by virtually pointing or physically touching objects with a part of human's body [AZ97].

The original formulation introduced by Fitts is given by [OGRP12]:

$$MT = a + b \cdot ID \quad (18)$$

where

$$ID = \log_2\left(\frac{2D}{W}\right) \quad (19)$$

MT is the average movement time spent when a pointing movement task is performed. a and b are constants determined by linear regression, they can be considered as reaction time taken when an action needs to be performed by the pointing device. The values of constants a and b depend on the performance of the pointing device as well as the tracking algorithm applied on the device. For instance, both Kinect and Leap Motion may be used for pointing, but the constants a and b are different for each of them. ID is the index of difficulty of the task that moves the pointer cursor from a start point into a target object with width W and D is the distance between the start point and the center of the target object.

Shannon modified Fitts's original version by changing the formula of ID as [MB92]:

$$ID = \log_2\left(\frac{D}{W} + 1\right) \quad (20)$$

In addition to guaranteeing the value of ID is always non-negative, the modified Fitts's Law also fits the measured data better than the original version [MB92].

3.6 Experiment Principle and Algorithm

Once an experiment is performed, the user will need to carry out a multi-directional dragging task as described in ISO 9241-400 [ISO10] which includes pre-defined number of trials. In each trial, there will be even objects placed along a circular track. The dragging tasks are arranged in target pairs. If the location of a source target is decided, the target object will be placed in the opposite side of the track.

Suppose there are n objects placed along the track and the objects are numbered from 0 to $n-1$ in ascend order. If the number of source object is i , then the number of target object j can be found by:

$$j = i + \left(\frac{n}{2} - 1\right) \pmod{n} \quad (21)$$

In the next round, the target object will be replaced as a new source object and the number of the new target object can be calculated by using the same formula as equation 21. Now that the system knows the number of the source object and the target object in each pair, the next step is to calculate the coordinate of each object according to their numbers.

Figure 7 shows an example with 16 objects on the track. Assume that the number of the source object is 0, the number of the target object can be computed and 7 is got as the result according to the equation 21. After the task for this round is done, now the number of the new source object becomes 7 and the number of the new target object becomes 14. The same process continues until the eighth round is done.

Equation 22 shows how the X axis coordinate of object with a number of k can be calculated according to the coordinate of the center of the track as well as its radius.

$$O_x = T_x + R \cdot \sin\left(\frac{360^\circ \cdot k}{n}\right) \quad (22)$$

where O_x and T_x are the X axis coordinates of the center of the object and the

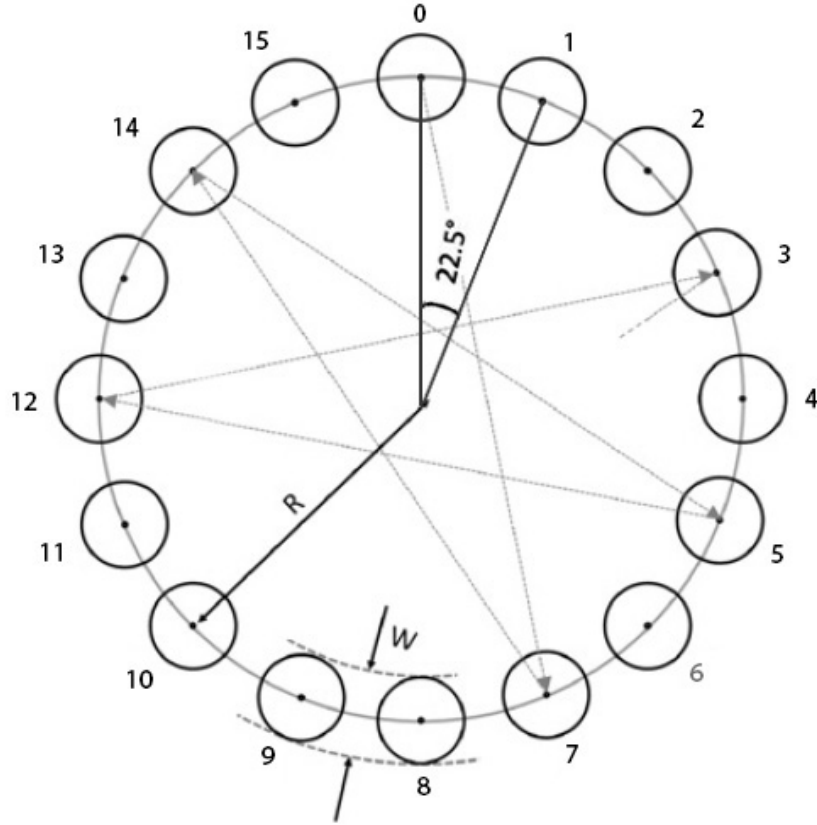


Figure 7: Experiment with 16 Objects.

center of the track respectively. R is the radius of the track and k is the number of the object. Similarly, the Y axis coordinate of the center of the object can be calculated by:

$$O_y = T_y + R \cdot \cos\left(\frac{360^\circ \cdot k}{n}\right) \quad (23)$$

where O_y and T_y are the Y axis coordinates of the center of the object and the center of the track respectively.

The coordinate of object k can be calculated as long as all the variables are known in equation 22 and 23. Refer to the Figure 7, if the coordinate of object 1 needs to be calculated, the only things need to be known are the coordinate of the center of

the track as well as its radius. Suppose that the coordinate of the center of the track is (500, 500) and its radius is 100 pixels. Since the amount of the objects is known as 16, the coordinate of the center of the object 1 can be calculated and (538, 592) is got as a result.

In order to carry out grab-release tasks during the experiment, the experimental platform has a functionality that allows any input devices to notify the system whether an object is movable or not. The system examine the movability of the object by monitoring the distance between the pointer cursor and the center of the object as well as the status whether the object is notified as clicked by the chosen gesture. Suppose that the radius of an object is r , the movability of the object can be divided into four cases (Table 1).

Case	Status_isClicked	Distance	Movable
1	no	greater than r	no
2	no	less than r	no
3	yes	greater than r	no
4	yes	less than r	yes

Table 1: Movability of Objects.

Case 1 and 2 show that no matter how far the cursor is away from the object, the object will not be movable as long as it is not notified as clicked. At the same time, case 3 and 4 show that the object is movable only when it is recognized as clicked and the cursor is located within the range of the object. It is worthwhile to note that the condition of case 4 does not always hold, a user may click on the object and then move the cursor into the range of the object. In this case, even the condition of case 4 is met, the object cannot be considered as movable since the actual click event happened before the cursor has been moved into the range of the object. Thus, a flag is placed to indicate the status of the object in the last frame. If the object is not clicked in the last frame, then the condition of case 4 holds.

4 Implementation

4.1 Design

The framework is applied as it is introduced in section 3.1 to our experimental platform. Leap Motion, Kinect and Mirametrix are integrated into the platform as the main input devices for testing purpose. The control agent is hand, arm or eyeball. The activity diagram (Figure 8) shows the activities that a user needs to perform in order to finish a complete round of experiment.

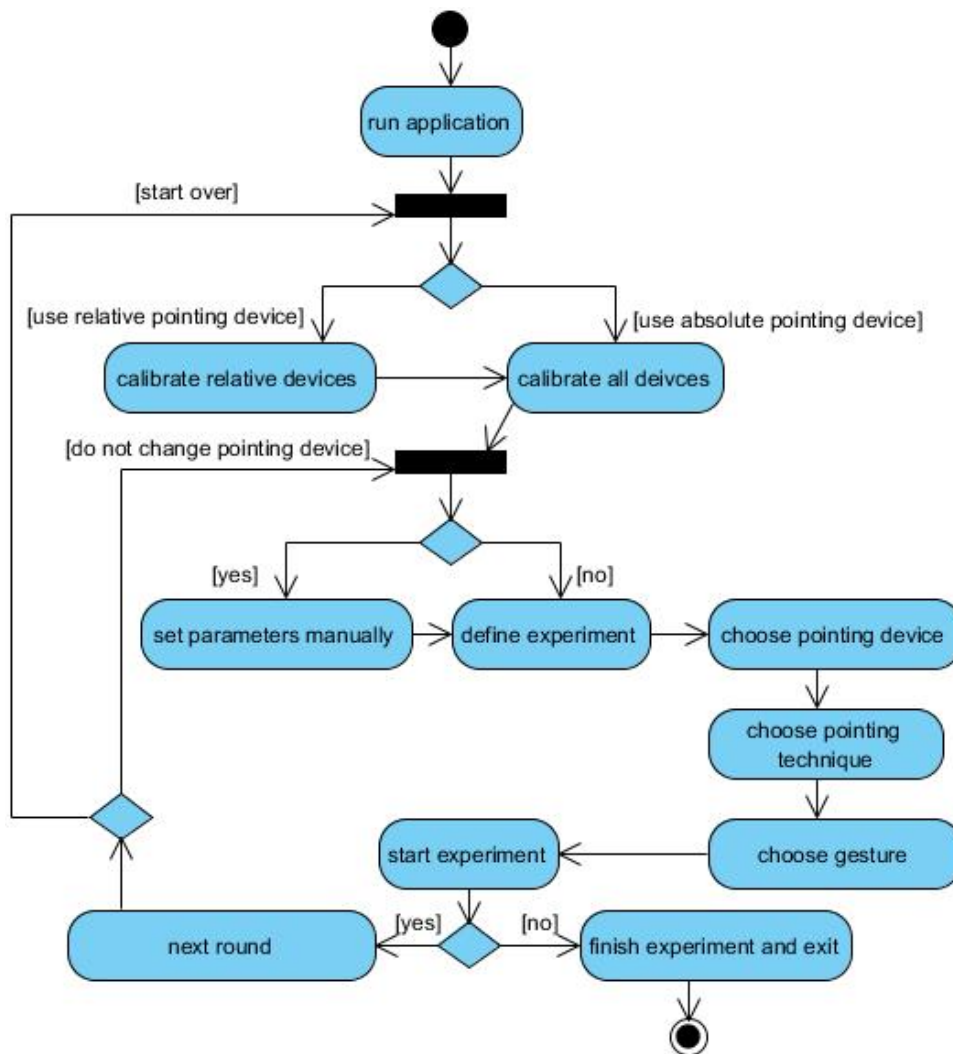


Figure 8: User Activity.

Once the application is run, the user who is using the platform has to determine whether the input pointing device offers a relative pointing measurement or an absolute pointing measurement. If the pointing device offers an absolute pointing measurement, then the user is required to perform a calibration task as discussed in section 3.4. The purpose of the calibration task is to calculate the value of 90 percentile d_{90} that allows a 90 percentages error rate while the user is trying to point at an expected spot. Otherwise, the user needs to perform another calibration task as discussed in section 3.2 so that the platform can transform the relative pointing input into the absolute pointing input according to the value of the offset between the motor space and the display space.

It is worthwhile to note that the results generated by the two calibrations are only valid for one person who interacts with a running pointing device in a couple of hours since every person has different habitual behavior and even the ability that a person can focus on a specific target may change from time to time. At the same time, the accuracy that a pointing device can offer is totally depending on its hardware specifications and the tracking algorithm it is applied with. Therefore, it is necessary to perform these two calibration tasks frequently to ensure the correctness of the results unless it is required to keep the same parameters.

When the pointing input is in absolute pointing mode either by the technique that the pointing device supports or by the calibration task the platform provides, the user can choose if he wants to perform the experiment task in adaptive pointing mode or smoothed pointing mode. If the former one is chosen, the user will have to manually set the parameters that the adaptive pointing technique relies on although the default set of parameters have been suggested by the platform. Otherwise, the user can skip this step if the smoothed pointing technique is chosen instead.

Before the experiment starts, the user has to decide the pointing device and the pointing technique that will be used during the experiment. Obviously, the decided pointing device has to be the one the user calibrated with. Finally, the gesture has to be decided to perform the grab-release tasks as the minimum requirement of the

experiment. The gesture can be provided by the devices differ from the one used for pointing purpose if needed.

Once a round is completed, the user can either quit the application or run the next round. If the user decides to continue the experiment with the same pointing device, he will have to start from the option to set the parameters manually or automatically. Otherwise, the user needs to start the whole process over again.

4.2 Hardware

The hardware is currently limited to a Leap Motion, a Kinect, a Mirametrix eye tracker and a 2.4G wireless mouse. Leap Motion, Kinect and Mirametrix are all video-based pointing devices.

There are four main reasons why Leap Motion, Kinect and Mirametrix are decided to be chosen as the pointing input devices. Firstly, they are widely used and can be found in markets easily. Secondly, they are connected to a computer through wired USB cables, one probability that the single may be lost in wireless connection does not need to be worried about although the nature of most video-based pointing devices can not be avoided. That is, noises may be caused by the environment factor. Thirdly, the control agents are different, they capture the motions of human's hands, body and eyeballs respectively. Finally, they are relative pointing devices, all of them need to perform the calibrations to get into the absolute pointing mode except Mirametrix since the manufacturer of Mirametrix has provided a functional calibration platform already [SAJ⁺11].

The mouse is used to replace the gesture function provided by other pointing input devices. Since video-based pointing devices cannot always recognize gestures correctly [KJA⁺13], it is necessary to have an input device with physical buttons to do the job for video-based pointing devices. A 2.4G wireless mouse is a proper choice. Compared with other input devices with physical buttons, mouse is relatively lighter and smaller. It is easy for a user to point with one hand and move around freely while holding a mouse with another hand. More importantly, the delay caused by

the wireless transmission can be ignored since 2.4G wireless communication is able to transfer small data with almost non-delay rate [XXH10].

4.3 Software and Libraries

The platform was developed in C# with Unity 3D(4.2.2f1) under Windows 7 Ultimate. The drivers used for Leap Motion, Kinect and Mirametrix were Leap Motion Controller 1.1.3, KinectSDK-1.8 and Mirametrix Tracker 2.5.1.152 respectively.

The main libraries used in the platform were UnityEngine [Tec14], Leap [LM14], FubiNET, System.Net [Cor14a] and System.Xml [Cor14b]. As a video game development tool, UnityEngine was used to display any visual graphics within the user interface, calibration tasks and experiment tasks. The GUI was also allowed to be displayed in a changeable frame rate and UnityEngine provided a functionality which could count the real time in each frame.

On the other hand, Leap and FubiNet were used to transform users' graphical motions into numerical data. For instance, when a user held his hand above a Leap Motion, the library could transform the position of the user's fingertip as a coordinate (X, Y, Z) according to the corresponding position between the device and user's fingertip in space. Since Mirametrix Tracker had provided the pointing function for Mirametrix already, it was unnecessary to import a library to handle the pointing and calibration tasks for Mirametrix. System.Net was used to create a socket which allowed Unity 3D and Mirametrix to exchange data.

The configurations of a experiment task did not need to be recorded by the user interface. Therefore, System.Xml was used to store the configurations into a XML file so that the old configurations would be replaced immediately once the newer ones were saved.

4.4 User Interface

The user interface mainly consists of five boxes. The layout of the user interface is shown in Figure 9.



Figure 9: Layout of User Interface.

The box in the upper left should be accessed first since the input should be ensured that it is in absolute pointing mode and the parameters for adaptive pointing and smoothed pointing have been measured. After the necessary calibrations are done, the user may set the parameters manually within the box under the former one (Figure 10).

Obviously, the platform is flexible, not only the performance of two pointing techniques can be adjusted by modifying the variables they rely on, but also the parameters of the experiment can be adjusted through the user interface. The box in the upper middle (Figure 11) is used to configure the color of object circulars and define the size of the circulars as well as the track that these circulars are placed on. Since users may desire to remain the experiment environment for several rounds, a button is offered to save the changes and another button is used to reset the configuration. The box on the right side provides a place for users to choose the pointing input device, the pointing technique and the gesture. The button with text "+" are

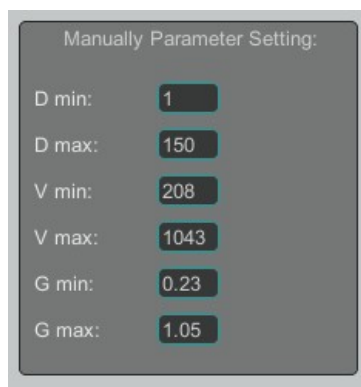


Figure 10: Manually Parameter Setting.

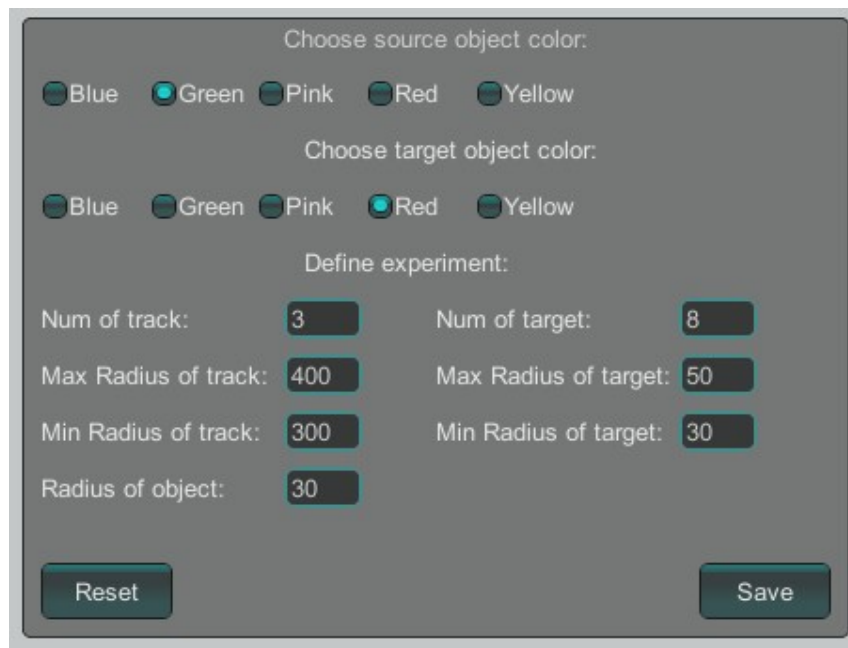
reserved for the future extension.

Eventually, the box at the bottom allows users to perform the decided experiment task. The second button within the box is also reserved for future extension, however, this experimental platform only supports for testing based on Fitts's Law at the moment.

4.5 Calibration

When a user performs the calibration task for relative pointing inputs. It is necessary to let the system know whether the user is pointing at either the left lower or the right upper of a display screen. This situation can not be realized by the system immediately since the system never know where the user is pointing at in the display space before the calibration is done. Thus, our solution is to let the user point at the left lower and the right upper of the screen in proper order. After a graphical notification circular is shown on the screen, if the user is pointing at a point and keep still for 3 seconds, then the truth that the user is pointing at the notified location can be firmly believed. The user does not need to be absolute still, however, any movement less than 50 pixels in two interfacing frame will be ignored.

If the calibration task for absolute pointing inputs is performed, a red crosshair



Choose source object color:

Blue Green Pink Red Yellow

Choose target object color:

Blue Green Pink Red Yellow

Define experiment:

Num of track: Num of target:

Max Radius of track: Max Radius of target:

Min Radius of track: Min Radius of target:

Radius of object:

Figure 11: Experiment Configuration.

(Figure 12) will be displayed in the middle of the screen. The user will need to point at the center of the crosshair to start the calibration task. Due to the nature of video-based pointing devices, it is impossible to match the actual point on the display screen with the location that a user wants to point at mentally even an absolute pointing mode is applied. Therefore, instead of caring about where his is going to point at physically, the user should concentrate on moving and keep the pointer cursor in the center of the crosshair. Since the user is able to recognize where the pointer cursor is, the system will start the calibration immediately once an adjacent cursor is detected.

Distinguished from other tracking techniques, eye tracking relies on the motions of eyes. It is extremely hard for a human to control the movement of a cursor while looking at another place if there is a deviation between the motor space and the display space. Thus, a similar solution is applied as it is used for adaptive pointing input. That is, the calibration will start in 3 seconds if the user aims at the center of the crosshair and non large movement is detected.

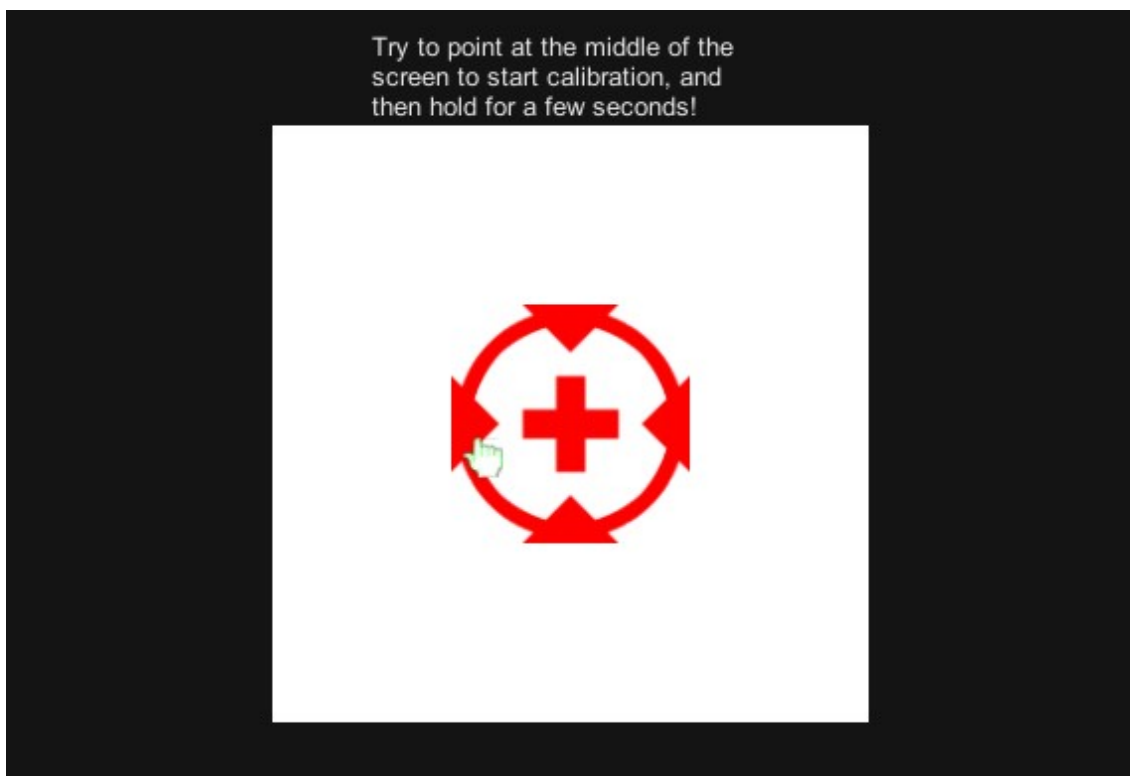


Figure 12: Crosshair for Calibration.

4.6 Experiment

Once the experiment starts, the system generates a number of tracks with different radius according to the parameters set in the user interface. Then each track will generate a number of objects and calculate their coordinates. The information of tracks and objects will be stored in a list. In order to ensure the fairness of the experiment, the order of displaying these tracks and objects is decided by the system randomly.

During the experiment, the source object and target object on the track are displayed in pairs. The user's task is to move the pointer cursor upon the source object, click the object with a defined gesture, drag the source object into the range of target object and release the gesture. Figure 13 shows the layout of the experiment in one round.

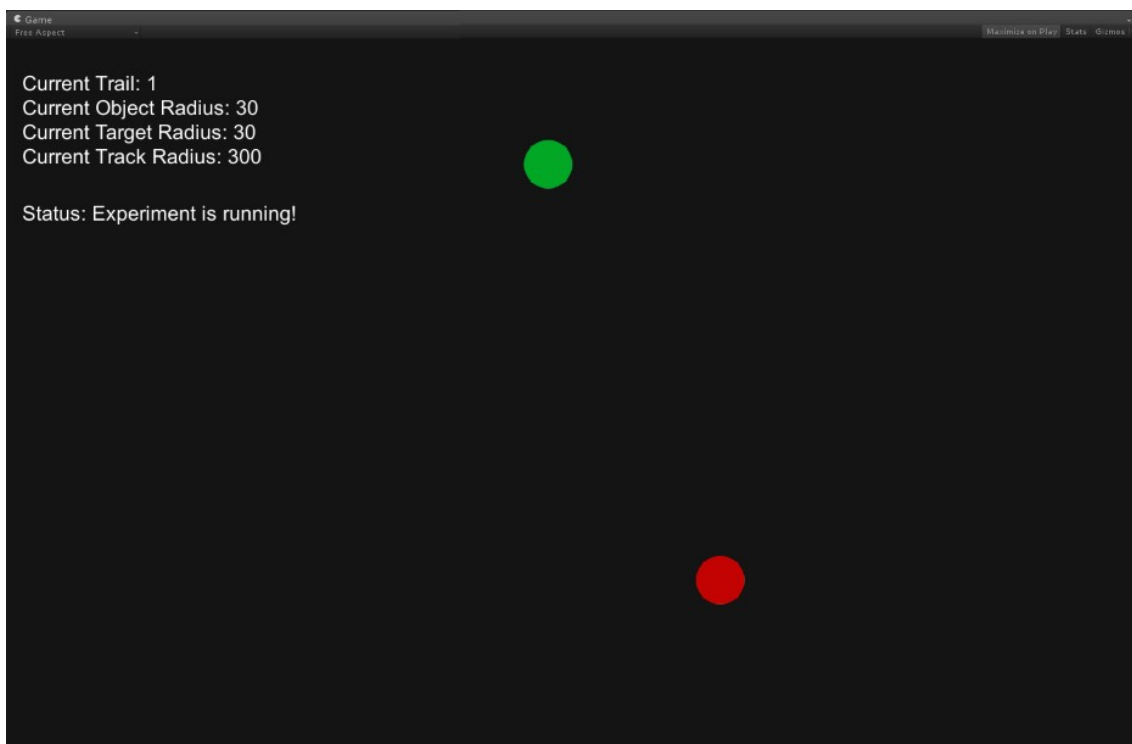
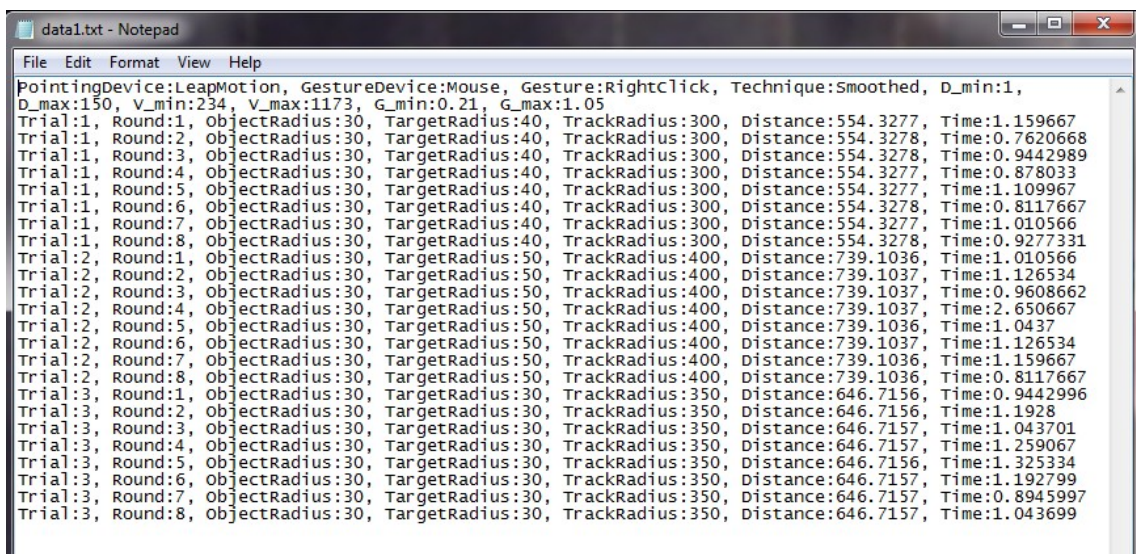


Figure 13: Layout of Experiment.

The parameters of the current track and objects are displayed as text in the upper right corner. If the source object is clicked, the system will start to count the timer. The counter will stop once the source object is dropped into the target object. At this moment, system will record the duration time, the distance between two objects, the settings of the experiment as well as the parameters of the objects and the track. Eventually, the system will log these records while each round in the experiment is processing so that those data can be analyzed by the user for any analysis purpose.

As an example, figure 14 shows the example data log of the experiment. The experiment was performed for three trials with 8 objects. First two lines show the settings of the experiment. The user used a Leap Motion to perform the experiment while a mouse was used to control the objects as the gesture. The parameters of the pointing technique were obtained by the system automatically since a smoothed pointing mode was activated. Meanwhile, the following lines show the data log generated by

the system in each round. The radius of the source objects were set by the user while the radius of the target objects and tracks were determined by the system randomly according to the information that the user gave in the user interface. The variable "Distance" indicated the distance between the present source object and the target object, it was fair for any round in one trial. Obviously, the conditions for every round in each trial were the same, but the duration time spent by the user to perform each task was different.



```

data1.txt - Notepad
File Edit Format View Help
PointingDevice:LeapMotion, GestureDevice:Mouse, Gesture:Rightclick, Technique:Smoothed, D_min:1,
D_max:150, V_min:234, V_max:1173, G_min:0.21, G_max:1.05
Trial:1, Round:1, ObjectRadius:30, TargetRadius:40, TrackRadius:300, Distance:554.3277, Time:1.159667
Trial:1, Round:2, ObjectRadius:30, TargetRadius:40, TrackRadius:300, Distance:554.3278, Time:0.7620668
Trial:1, Round:3, ObjectRadius:30, TargetRadius:40, TrackRadius:300, Distance:554.3278, Time:0.9442989
Trial:1, Round:4, ObjectRadius:30, TargetRadius:40, TrackRadius:300, Distance:554.3277, Time:0.878033
Trial:1, Round:5, ObjectRadius:30, TargetRadius:40, TrackRadius:300, Distance:554.3277, Time:1.109967
Trial:1, Round:6, ObjectRadius:30, TargetRadius:40, TrackRadius:300, Distance:554.3278, Time:0.8117667
Trial:1, Round:7, ObjectRadius:30, TargetRadius:40, TrackRadius:300, Distance:554.3277, Time:1.010566
Trial:1, Round:8, ObjectRadius:30, TargetRadius:40, TrackRadius:300, Distance:554.3278, Time:0.9277331
Trial:2, Round:1, ObjectRadius:30, TargetRadius:50, TrackRadius:400, Distance:739.1036, Time:1.010566
Trial:2, Round:2, ObjectRadius:30, TargetRadius:50, TrackRadius:400, Distance:739.1037, Time:1.126534
Trial:2, Round:3, ObjectRadius:30, TargetRadius:50, TrackRadius:400, Distance:739.1037, Time:0.9608662
Trial:2, Round:4, ObjectRadius:30, TargetRadius:50, TrackRadius:400, Distance:739.1037, Time:2.650667
Trial:2, Round:5, ObjectRadius:30, TargetRadius:50, TrackRadius:400, Distance:739.1036, Time:1.0437
Trial:2, Round:6, ObjectRadius:30, TargetRadius:50, TrackRadius:400, Distance:739.1037, Time:1.126534
Trial:2, Round:7, ObjectRadius:30, TargetRadius:50, TrackRadius:400, Distance:739.1036, Time:1.159667
Trial:2, Round:8, ObjectRadius:30, TargetRadius:50, TrackRadius:400, Distance:739.1036, Time:0.8117667
Trial:3, Round:1, ObjectRadius:30, TargetRadius:30, TrackRadius:350, Distance:646.7156, Time:0.9442996
Trial:3, Round:2, ObjectRadius:30, TargetRadius:30, TrackRadius:350, Distance:646.7156, Time:1.1928
Trial:3, Round:3, ObjectRadius:30, TargetRadius:30, TrackRadius:350, Distance:646.7157, Time:1.043701
Trial:3, Round:4, ObjectRadius:30, TargetRadius:30, TrackRadius:350, Distance:646.7157, Time:1.259067
Trial:3, Round:5, ObjectRadius:30, TargetRadius:30, TrackRadius:350, Distance:646.7156, Time:1.325334
Trial:3, Round:6, ObjectRadius:30, TargetRadius:30, TrackRadius:350, Distance:646.7157, Time:1.192799
Trial:3, Round:7, ObjectRadius:30, TargetRadius:30, TrackRadius:350, Distance:646.7157, Time:0.8945997
Trial:3, Round:8, ObjectRadius:30, TargetRadius:30, TrackRadius:350, Distance:646.7157, Time:1.043699

```

Figure 14: Output of Experiment.

5 Evaluation

The experimental platform was evaluated by reviewing with several experts in HCI and getting feedback from them. 4 volunteers (2 males, 2 females) were served as experts. They had at least on year of research experience in HCI. The ages of these experts were from 28 to 33 years old. Three of the experts were right-handed while the other one was left-handed and they used computers on a daily basis. Two out of four had pointing experience with Leap Motion, Kinect and Mirametrix. All of them were used to playing 3D games.

5.1 Apparatus

The review was conducted on an Intel i5 2.6GHz desktop computer with a 50" Samsung television. The input devices were limited to a Leap Motion 3D Controller, a Kinect for Windows, a Mirametrix S2 Eye Tracker and a Logitech Wireless Mouse M560.

The TV was hung on the wall 110 centimeters from the ground. The Kinect was placed on the bottom of the TV and all experts were required to stand 2 meters away from the Kinect because Kinect sensor required a minimum 1.4 meters space between the device and the object in order to detect a person's motions accurately [FPT12]. Due to the nature of Mirametrix, it needs to catch the movement of a relative smaller object and requires more accurate graphical recognition than Kinect. Thus, the Mirametrx was placed in front of the experts where the Mirametrix is 60 centimeters away from the experts and 1.4 meters away from the television respectively. Meanwhile, since Leap Motion was used to catch the motions of the experts, it was placed on both side where the experts could hold their hands above the device naturally. The location of mouse was not fixed since the experts would need to hold it in their hand and click the right button.

The default frame per second(FPS) of Kinect is 30 while the FPS of Leap Motion is 60 [FPT12][PAC13]. Therefore, the FPS was limit to 30 for both devices so that

the Leap Motion and the Kinect could be reviewed fairly. Unlike Kinect and Leap Motion, Mirametrix tracks the movement of human's eyes. Instead of dragging a pointer cursor from one place to another place all the way along the actual path of human's motions, Mirametrix monitors the movement of human's eyes to control the cursor from point to point jumpily. The Mirametrix did not care about how the path went, that was the reason why the FPS of it was limited to 10.

The Leap Motion and Kinect can be operated by either the left hand or the right hand. If an expert decide to use Leap Motion or Kinect as a pointing device and use the right button of the mouse as the gesture, he has to pointing with one hand and control the mouse with another hand. If Mirametrix is used, then which hand should be used to control the mouse depends on expert's preference.

5.2 Procedure

Four experts were asked to test the experimental platform with different pointing devices. Each expert was given 5 minutes to get familiar with Leap Motion, Kinect and Mirametrix through a simple pointing task. The test was performed in 6 turns. Each expert tested the experimental platform starting from the calibration tasks and ending by the experiment tasks.

Both adaptive pointing and smoothed pointing were tested with each pointing device in proper order and each gesture was used twice. Except showing how the gestures operated, no guidance was given during the test.

The experimental platform required the experts to start with choosing a pointing device and performing a calibration task according to the input type that the pointing device supported by default. The parameters would be given automatically through the calibration task for absolute input if a smoothed pointing mode was selected. Otherwise, the experts were required to set the parameters manually.

After the experts had decided the color of the source object and the target object, they configured the variables for the experiment tasks such as the maximum and

minimum radius of the tracks and target objects as well as the number of trials and objects. Once the configurations for the experiment task and pointing technique were ready, they chose a proper pointing device, a pointing technique and a gesture to run the experiment task.

During the experiment task, the experts were asked to drag a source object into a target object for several rounds. The size of target objects and tracks were different in each trial. Once the experiment task was done, the output data would be available.

5.3 Result

At the end of the test, the experts were asked to offer some suggestions and describe the impression of the experimental platform in several respects such as the major flaws experienced with pointing devices, gestures, user interface, calibration tasks, experiment tasks and outputs.

All the experts complained that pointing with one hand and making the gesture with the same hand affected the correctness of the pointing experience. Using the button of the wireless mouse as a gesture was the most preferred choice.

Concerning the user interface, three out of four experts suggested to insert an box to the user interface that allowed the users to input information of themselves. Two out of four experts wanted to have the configuration of experiment tasks in centimeters and inches as an extra choice. One expert had trouble with figuring out the right order of processing the boxes in the user interface.

Although all experts were satisfied with the calibration tasks, one expert complained that the objects were overlapped with the text displayed in the experiment task. Meanwhile, another expert suggested to having the round information so that the user could realize how many rounds remained during the experiment task.

On the other hand, two out of four experts wanted the output of size and position data to be generated in centimeters. One expert wanted to have the real time included in the output and suggested that the format of the output could be different

for European Standard and American Standard. Finally, another expert suggested to add a variable selection function for the output.

5.4 Discussion

As expected, the experimental platform was able to guide the experts to complete the experiment tasks and generate proper output for analysis purpose. All the pointing devices used in the test were able to be calibrated and generate expected output data. However, pointing devices like Leap Motion and Kinect were not yet suitable for extended usage during the experiment task due to the nature of video-based pointing devices. For instance, if a user is dragging a source object and the system suddenly lose the gesture recognition, the duration time of the dragging task will no longer be valuable references. Thus, it is necessary to have a device with physical button to be used as a click event unless the user desire to count the correctness of gesture recognition as a part of the analysis.

The user interface provided the basic functions for the experts to access the modules in the experimental platform. The system could be even easier to deploy in experiments if the user interface could provide some text fields for users to input their personal information such as hand preference, ethnic group, weight, height, glasses or length of the index finger since the experiment conductor would not need to gather these data with separate questionnaires. Those information could be consulted as a factor of the data log. At the same time, some limitation could be applied to the configuration for the experiment task. For example, the radius of the source object can never be greater than the radius of the target object and the location of objects can not be exceeded the displayable space of the screen according to the sum of the radii of tracks and objects.

The textual and graphical information provided by the experiment task were able to guide the experts to perform the tasks in the right way. Nevertheless, the layout of the experiment task could be improved so that the information could be displayed correctly in any case. Tracks and objects could be bounded into a limited area in

order to avoid the overlap of objects and textual information. On the other hand, the textual information could be even more informative if it could include the number of rounds to let users realize the number of remain tasks that will be performed.

The output generated all the necessary data related to the experiment task. The output could be even more humanized if users was able to decide which variable should be involved in the output. Eventually, instead of using a comma as the separator of different variables in the output, it was useful to use a vertical bar as the separator since comma was added to numbers after every third digit from right to left in some countries while other countries use a dot and this situation might mislead the separator.

6 Conclusions and Further Work

In this thesis, a configurable framework was applied to the design of an experimental platform. A composite synthesis architecture comprising two enhanced pointing techniques and Fitts's Law were proposed and used as a conceptual model. A prototype application experimental platform was then implemented in Unity 3D using C# as the programming language. The main results of the thesis are discussed in section 6.1 and some additional features and improvements are suggested in section 6.2.

6.1 Conclusions

The entire process from architectural design to the actual implementation phase was repeatable. The two-point calibration algorithm should best be regarded as a normalized calibration technique for most video-based pointing devices while a nine-point calibration technique was supplied for more precise pointing devices (i.e. Eye Tracking System). The calibrated absolute pointing input was made to be enhanced either by adaptive pointing or smoothed pointing technique. Adaptive pointing worked better when a more flexible adjustment needed to be applied during the experiment task while smoothed pointing worked better if a pure absolute pointing experience was required as the key condition for the measurement. Meanwhile, the contained variables of a well-known model Fitts's Law were consulted as the principle of the experiment algorithm. The experiment algorithm was able to decide the order of the tasks and the coordinate of each objects.

The implementation phase of the experimental platform succeeded relatively well. The libraries provided by Unity Engine, Leap and Fubi were sufficient for the development. All the modules worked in proper order. A clear path from calibration task, via parameter configuration to the experiment task was proposed for users to perform a complete experiment. As a result, the format of the output was defined to store all the necessary variables which were valuable for the future analysis purpose.

The experimental platform met the essential needs of the professional pointing analysis. It was easy for users to operate the platform without any verbal guidance. Although the functions of the experimental platform worked well as expected, there were still some user experience issues which remained for future efforts.

6.2 Future Research and Extensions

Although the overall concept of experimental platform is already in its final stage, the platform itself still needs a function which allows user to load any pointing modules with an associated pointing device. If a new pointing technique or experiment task is available, they should be loaded as modules dynamically as well.

A new pointing technique should be introduced to overcome the factor of smoothed pointing that the algorithm of smoothed pointing holds only under the condition that a 6/6 vision is assumed by default.

It will be interesting to include a feature which can evaluate and compare the performance between two solutions since users are responsible for analyzing the output data on their own so far.

References

- 3Dc14 3Dconnexion, I., Spaceball 5000, February 2014. URL <http://www.cs-software.com/hardware/3Dconnexion/spaceball.html>.
- AZ97 Accot, J. and Zhai, S., Beyond fitts' law: Models for trajectory-based hci tasks. *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, CHI '97, New York, NY, USA, 1997, ACM, pages 295–302.
- Cas06 Casamassina, M., Wii controllers: Unlocking the secrets, <http://www.ign.com/articles/2006/07/15/wii-controllers-unlocking-the-secrets/>, July 2006.
- Cor14a Corporation, M., System.net namespace, 2014. URL [http://msdn.microsoft.com/en-us/library/system.net\(v=vs.110\).aspx](http://msdn.microsoft.com/en-us/library/system.net(v=vs.110).aspx).
- Cor14b Corporation, M., System.xml namespace, 2014. URL [http://msdn.microsoft.com/en-us/library/system.xml\(v=vs.110\).aspx](http://msdn.microsoft.com/en-us/library/system.xml(v=vs.110).aspx).
- CS10 CyberGlove Systems, L., Cyberglove iii, 2010. URL <http://www.cyberglovesystems.com/products/cyberglove-III/overview>.
- CVC12 Cerrolaza, J. J., Villanueva, A. and Cabeza, R., Study of polynomial mapping functions in video-oculography eye trackers. *ACM Trans. Comput.-Hum. Interact.*, 19,2(2012), pages 10:1–10:25.
- CVPC07 Casiez, G., Vogel, D., Pan, Q. and Chaillou, C., Rubberedge: Reducing clutching by combining position and rate control with elastic feedback. *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology*, UIST '07, New York, NY, USA, 2007, ACM, pages 129–138.
- DM07 Delage, E. and Mannor, S., Percentile optimization in uncertain markov decision processes with application to efficient exploration. *Proceedings*

of the 24th International Conference on Machine Learning, ICML '07, New York, NY, USA, 2007, ACM, pages 225–232.

- DRBS90 Driver, J., Read, R. L., Blough, E. and Seah, K., An evaluation of the polhemus, spaceball, and mouse devices for 3-d cursorpositioning. Technical Report, Austin, TX, USA, 1990.
- Ett11 Etter, W., Ir camera system, <http://williametter.com/portfolio/projects/wii-ir-camera-system/>, February 2011.
- FPT12 Francese, R., Passero, I. and Tortora, G., Wiimote and kinect: Gestural user interfaces add a natural third dimension to hci. *Proceedings of the International Working Conference on Advanced Visual Interfaces, AVI '12*, New York, NY, USA, 2012, ACM, pages 116–123.
- GM12 Gallo, L. and Minutolo, A., Design and comparative evaluation of smoothed pointing: A velocity-oriented remote pointing enhancement technique. *Int. J. Hum.-Comput. Stud.*, 70,4(2012), pages 287–300.
- ISO10 ISO, Ergonomics of human–system interaction – part 400: Principles and requirements for physical input devices. Iso, International Organization for Standardization, Geneva, Switzerland, June 2010.
- KGDR09 König, W. A., Gerken, J., Dierdorf, S. and Reiterer, H., Adaptive pointing — design and evaluation of a precision enhancing technique for absolute pointing devices. *Proceedings of the 12th IFIP TC 13 International Conference on Human-Computer Interaction: Part I, INTERACT '09*, Berlin, Heidelberg, 2009, Springer-Verlag, pages 658–671.
- KHW95 Kessler, G. D., Hodges, L. F. and Walker, N., Evaluation of the cyberglove as a whole-hand input device. *ACM Trans. Comput.-Hum. Interact.*, 2,4(1995), pages 263–283.
- KJA⁺13 Kosunen, I., Jylha, A., Ahmed, I., An, C., Chech, L., Gamberini, L., Cavazza, M. and Jacucci, G., Comparing eye and gesture pointing to

- drag items on large screens. *Proceedings of the 2013 ACM International Conference on Interactive Tabletops and Surfaces, ITS '13*, New York, NY, USA, 2013, ACM, pages 425–428.
- Lim13 Limited., W. M. N., How to place the wii balance board, 2013. URL <http://www.supercheats.com/wii/how-to/how-to-place-the-wii-balance-board>.
- LM14 Leap Motion, I., Api overview, 2014. URL https://developer.leapmotion.com/documentation/csharp/devguide/Leap_Overview.html.
- MB92 MacKenzie, I. S. and Buxton, W., Extending fitts' law to two-dimensional tasks. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '92*, New York, NY, USA, 1992, ACM, pages 219–226.
- OGRP12 Olafsdottir, H. B., Guiard, Y., Rioul, O. and Perrault, S. T., A new test of throughput invariance in fitts' law: Role of the intercept and of jensen's inequality. *Proceedings of the 26th Annual BCS Interaction Specialist Group Conference on People and Computers, BCS-HCI '12*, Swinton, UK, UK, 2012, British Computer Society, pages 119–126.
- PAC13 Potter, L. E., Araullo, J. and Carter, L., The leap motion controller: A view on sign language. *Proceedings of the 25th Australian Computer-Human Interaction Conference: Augmentation, Application, Innovation, Collaboration, OzCHI '13*, New York, NY, USA, 2013, ACM, pages 175–178.
- PB02 Pullen, K. and Bregler, C., Motion capture assisted animation: Texturing and synthesis. *ACM Trans. Graph.*, 21,3(2002), pages 501–508.
- PRS+94 Preece, J., Rogers, Y., Sharp, H., Benyon, D., Holland, S. and Carey, T.,

Human-Computer Interaction. Addison-Wesley Longman Ltd., Essex, UK, UK, 1994.

- SAJ⁺11 Skovsgaard, H., Agustin, J. S., Johansen, S. A., Hansen, J. P. and Tall, M., Evaluation of a remote webcam-based eye tracker. *Proceedings of the 1st Conference on Novel Gaze-Controlled Applications*, NGCA '11, New York, NY, USA, 2011, ACM, pages 7:1–7:4.
- SxQlH07 Shi-xu, S., Qi-lun, Z. and Han, H., A fast algorithm for real-time video tracking. *Proceedings of the Workshop on Intelligent Information Technology Application*, IITA '07, Washington, DC, USA, 2007, IEEE Computer Society, pages 120–124.
- Tec14 Technologies, U., Unity scripting reference, 2014. URL <http://docs.unity3d.com/Documentation/ScriptReference/index.html>.
- WBA⁺13 Wojtczuk, P., Binnie, D., Armitage, A., Chamberlain, T. and Giebeler, C., A touchless passive infrared gesture sensor. *Proceedings of the Adjunct Publication of the 26th Annual ACM Symposium on User Interface Software and Technology*, UIST '13 Adjunct, New York, NY, USA, 2013, ACM, pages 67–68.
- XXH10 Xiaolin, J., Xuemai, G. and Haijun, L., Based on 2.4g wireless communications system design. *Proceedings of the 2010 Second International Conference on Networks Security, Wireless Communications and Trusted Computing - Volume 02*, NSWCTC '10, Washington, DC, USA, 2010, IEEE Computer Society, pages 553–556.