VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wānanga o te Ūpoko o te Ika a Māui

School of Engineering and Computer Science
Te Kura Mātai Pūkaha, Pūrorohiko

PO Box 600
Wellington
New Zealand

Tel: +64 4 463 5341
Fax: +64 4 463 5045
Internet: office@ecs.vuw.ac.nz

Understanding SDAZ - Mouse vs. Touch vs. 3D Gestures

Jiaheng Wang

Supervisors: Craig Anslow, Stuart Marshall

Submitted in partial fulfilment of the requirements for Bachelor of Engineering with Honours in Software Engineering.

Abstract

Igarashi and Hinckley [7] developed a novel technique for navigating large documents called Speed-dependent Automatic Zooming (SDAZ). The SDAZ algorithm allows the zoom of the document to scale automatically depending on the speed the user is scrolling. Cockburn and Savage [3, 11] conducted user studies on SDAZ with traditional scrolling and zooming techniques, and found that SDAZ significantly improved users' performance. Previous SDAZ studies only considered single input implementations (*mouse/joystick* [7] and *mouse* [3, 11]), but no research has considered comparing multiple input devices. This paper presents a controlled user experiment comparing SDAZ for mouse, touch, and 3D mid-air gesture input, and evaluated the efficiency and effectiveness for navigating maps and text documents. Results show that the performance of participants (N=30) were slower with SDAZ for all input devices but they preferred the technique subjectively via a NASA Task Load Index assessment. With further familiarity and refinement of the implementation for non-mouse input devices, SDAZ may surpass traditional navigation techniques.

Acknowledgments

I would like to express my gratitude to my supervisor Craig for all the support, advice, and encouragement he has given me over this past year. This project would not have gotten this far without him. I very much look forward to working with you again in the future.

Contents

Chapter 1

Introduction

Due to the widespread use of computers and smartphones today, many document navigation techniques are standardized, and users often expect the same behaviour across different devices. However that is not always the case especially when interacting with the same document on different applications and operating systems. This project looked at applying an older navigation technique initially proposed for mouse to the relatively new and not yet widespread input method of 3D mid-air gestures

To improve navigation within large documents Igarashi and Hinckley [7] developed a novel technique for large documents called Speed-dependent Automatic Zooming (SDAZ)[1]. SDAZ allows the zoom of the document to scale automatically depending on the speed the user is scrolling. Cockburn and Savage [3, 11] conducted formal user evaluations of SDAZ and found users performed scrolling tasks faster with SDAZ. Users also subjectively preferred SDAZ over traditional navigation techniques.

This project implemented SDAZ for 3D mid-air gestures and compare it against mouse and touch navigation input for maps and text documents. The purpose was to understand what input mode (mouse, touch, and 3D mid-air gestures) is more efficient in terms of time and errors, and perceived effective from participants for SDAZ.

Both Igarashi and Cockburn's studies were conducted in the early 2000s and were performed with only mouse input which was the dominant input mechanism at the time. With the ubiquitousness of computers and smartphones nowadays, users are much more familiar with mouse and touch input as well as well-grounded and widespread navigation techniques for panning, scrolling, and zooming. As far as I am aware, SDAZ has never been applied to newer input modes notably touch or 3D mid-air hand gestures before.

To explore these more recent input modes I implemented SDAZ for mouse, touch, and 3D mid-air gestures. With which I conducted a user experiment to compare mouse navigation input for maps and text documents with Non-SDAZ and SDAZ against touch and 3D gestures. The purpose was to understand what input mode (mouse, touch, and 3D mid-air gestures) is most efficient in terms of time and errors, and perceived effectiveness from participants for SDAZ in document navigation. The results showed that navigation was slower with 3D mid-air gestures but performed fastest for mouse and then touch input. SDAZ was slightly slower for mouse and touch and slightly faster for 3D mid-air gestures. Qualitative results favoured SDAZ.

[1] `https://www.youtube.com/watch?v=fOJTna3sGzE`

Chapter 2

Related Work

2.1 Speed-Dependent Automatic Zooming

An alternative document scrolling technique is the rate-based scrolling [13]. This can be accessed on modern machines by pressing the mouse wheel and displacing the mouse cursor position. The scrolling speed is proportional to the mouse position displaced after the technique was activated. The problem with rate-based scrolling is an upper limit on the practical speed one can scroll at, as excessive speed can cause disorientation in the user due to extreme visual flow.

Igarashi and Hinckley [7] found that it is difficult to navigate large documents efficiently. Even small movements of the scroll bar causes sudden jumps in the document which disorientates users. To solve the issues of rate-based scrolling they developed Speed-Dependent Automatic Zooming (SDAZ) to navigate large documents more efficiently [7]. SDAZ is a novel navigation technique and combines rate-based scrolling and zooming. The aim of SDAZ allows users to locate distant targets in large documents without having to constantly change between zooming and moving, and without becoming disorientated by fast scrolling.

The concept behind SDAZ is that the zoom level is adjusted automatically depending on the rate which the user scrolls, thus, speed-dependent automatic zooming. The system zooms out when the user scrolls fast, and zooms back in when the user slows down. Such that:

$$scale = constant/speed$$

In implementation there are various problems with such a formula. The inverse of a linear line begins with a sudden drop and then flattens out. Such a change in zoom would greatly disorientate users. This was revised such that the zoom is an exponential of the speed:

$$scale = s0^{(dy-d0)(d1-d0)}$$

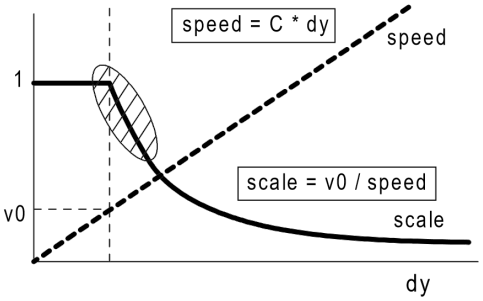


Figure 2.1: Igarashi's original mapping. Linear inverse of speed

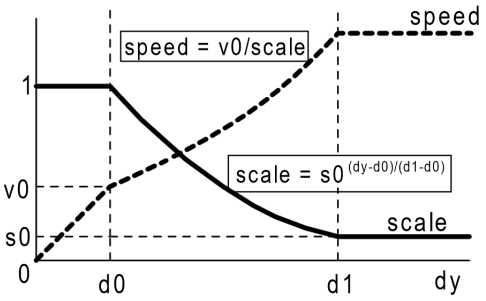


Figure 2.2: Igarashi's revised mapping. Exponential of speed

Another issue in implementation is the case where the user reverses suddenly. The speed in this case would go from positive to negative (or vice verse), causing a sudden zoom-in as the speed reaches zero, followed by a sudden zoom-out as the speed picks up again. This was solved by introducing a limit on how fast the scale changes (Figure 2.1). This limit was only applied to zooming in, as limit the rate which documents zoomed out would reintroduce the extreme visual flow that SDAZ aims to eliminate.

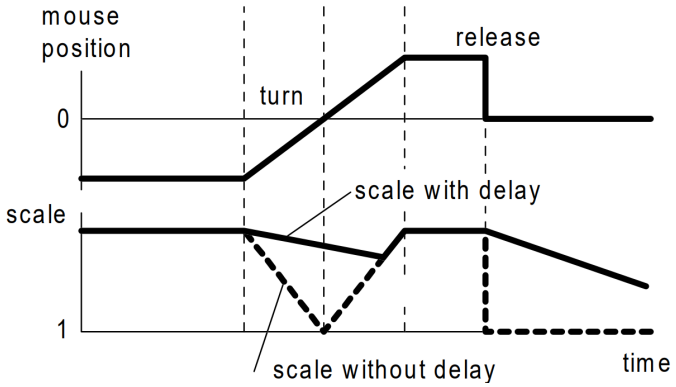


Figure 2.3: Igarashi's zoom-in delay

2.2 SDAZ User Evaluation

Igarashi and Hinckley applied SDAZ to several example applications. Web browser, map viewer, image browser, dictionary, and sound editor. They performed informal usability studies with their prototypes to identify the strengths and limitations of SDAZ for web browsing and map viewing, and observed the participants' reactions. They had seven test participants all of moderate experience with computers. The participants were required to complete tasks with both SDAZ and non-SDAZ techniques. They observed that SDAZ performed poorly on applications other than web browser and map viewer. This is due a lack of spatial relationship in the other applications.

2.2.1 Web Browser

The web browser implementation had a pink slider appearing when the mouse button is held. The document begins to scroll when the user moves the mouse while holding down

the button, with the displacement between the initial position and current position specifying the scroll speed. The document zooms out automatically as the speed increases, returning to normal when the user releases the mouse button. Headings in the document (chapter titles) do not zoom out, serving as the "landmarks" of the document. In the usability study the time taken to complete tasks was approximately equal for both SDAZ and non-SDAZ, but six out of the seven indicated that they preferred SDAZ.

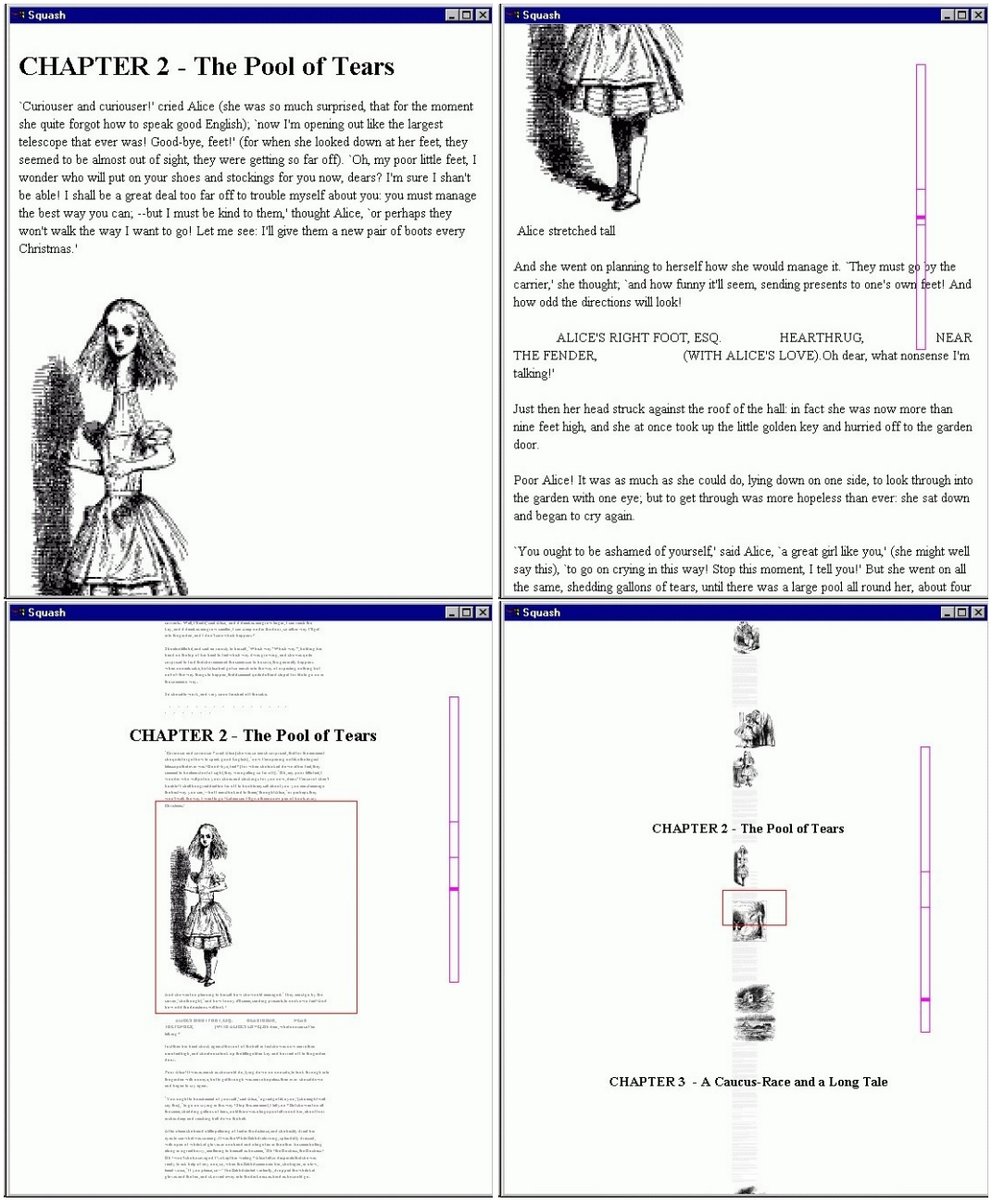


Figure 2.4: Igarashi's web browser implementation at four different scroll speeds

2.2.2 Map Viewer

The map viewer consisted of two input devices, a mouse and a joystick, on a noise function generated map. Joysticks were chosen for their self-centering effect [13]. On the mouse,

the displacement of the mouse from where the dragging operation started and the current position specifies the direction and speed of the camera. As the speed increases, the map automatically zooms out, returning to the original when the user releases the mouse button. On the joystick, the speed which the camera moves depends on how far the user tilts the stick. One problem with the joystick was that first-time users tilted the joystick to the maximum, causing sudden changes in the camera. Users had to learn to use smaller motions when operating the joystick. The task completion time in the usability study was more mixed compared to web browser. As navigating the map is more difficult, test participants utilized a range of differing strategies. Qualitative evaluations were also mixed. Four preferred SDAZ while three did not.

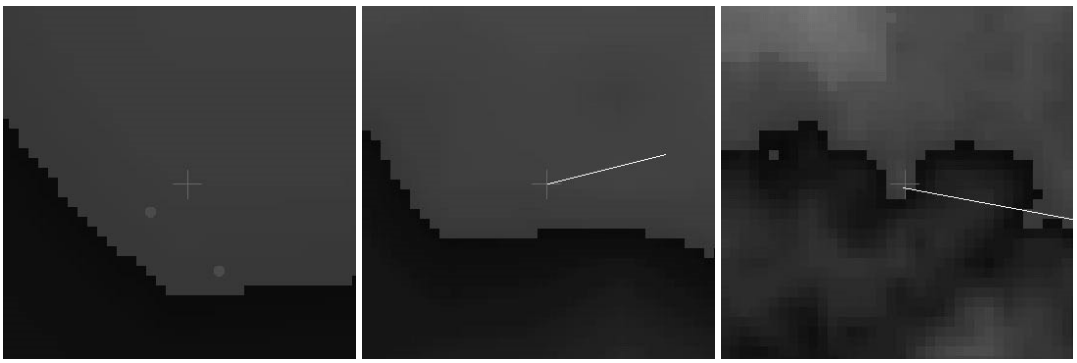


Figure 2.5: Igarashi's map viewer implementation at three different scroll speeds

2.2.3 Formal Evaluation

Igarashi and Hinckley's original proposal only had an informal evaluation of SDAZ which meant to provide initial impressions of the technique rather than accurate measures of the effectiveness. Cockburn and Savage [3, 11] conducted formal evaluations of the technique with positive results. SDAZ outperformed non-SDAZ techniques on both qualitative and quantitative evaluations. On average, SDAZ was 22% faster on text document tasks, 43% faster on maps, and was given higher subjective scores by the participants.

There are several key differences between Igarashi and Savage's implementations. Savage's system was written in C and OpenGL, which allowed for higher frame-rates more fluid animations compared to Igarashi's Java implementation. There is also a slight change in the SDAZ algorithm. Savage used a linear relationship between the scroll speed and zoom level, whereas Igarashi's was an inverse/exponential relationship.

Differences in the experimentation methodology are also contributors to the discrepancy between Savage and Igarashi's results. All twelve of Savage's participants were graduate level Computer Science students, or "expert", with all but three regularly played computer games. Whereas Igarashi's participants were "average" or "good", and four of the seven reported that they played computer games at a frequency of "sometimes" or higher. Savage and Cockburn also utilized an actual map of their city, whereas Igarashi and Hinckley's map was generated from a noise function.

2.3 Semi-Automatic Zooming

Kraz et al. [9] conducted a study of 2D tilt orientation based input on mobile devices with high resolution maps. By using the device's tilt as an input, users' fingers do not occlude the screen allowing the user to see the entirety of the map at all times. They implemented

a new technique called Semi-Automatic Zooming (SAZ) and compared it to a SDAZ implementation. SAZ differs from SDAZ in that users can opt to manually set the zoom level by using a slide on the touch screen of the device, returning to automatic zooming when the user releases the slider.

Their study revealed that SAZ performed significantly better than SDAZ. They found that SAZ is comparable in performance and usability to a standard multi-touch map interface. However, they did not compare their technique with mouse input.

2.4 3D Mid-air Gestures Navigation

There is not yet a widely accepted standard set of 3D mid-air gesture commands compared with touch interfaces [12]. This is due to mid-air gesture recognition devices not experiencing the same ubiquitousness of mouse and touch devices. However, with low cost devices such as Microsoft Kinect and Leap Motion we are starting to see more applications supporting 3D gestures.

Karam and Schraefel [8] created a scheme to classify gestures in HCI. Aigner et al. [1] extended this scheme to define five classifications of 3D mid-air gestures. These classifications include:

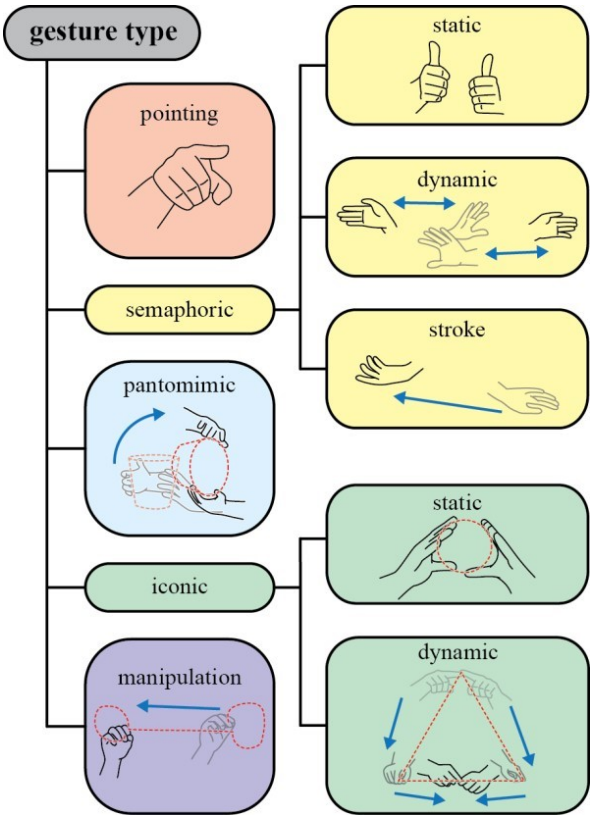


Figure 2.6: Aigner et al. [1] Classification of gestures

Pointing is used to indicate objects and directions. It does not necessarily involve a stretched index finger. Other forms of pointing are also accepted. Naturally pointing is most suitable for selection actions.

Semaphoric gestures are movements with specific meanings derived from social symbols or etiquette. For example thumbs-up is widely accepted to mean "okay". Semaphoric

gestures are further divided into three sub-categories. **Static semaphorics** are identified by a specific hand posture such as the previous thumbs-up example. **Dynamic semaphorics** uses temporal aspects through continual movement. A circular hand motion could mean "rotate", while repeatedly flicking the hand sideways could mean "no". **Semaphoric strokes** are similar to dynamic semaphorics, but consists of a singular stroke. Such as swiping on a touch device to mean "previous page" or "next page".

Pantomimic gestures are used to demonstrate the task to be performed by miming the actions. For example grabbing an object, moving it, then releasing it again.

Iconic gestures communicate information about objects, such as shapes, sizes, and motion paths. Iconic gestures and further subdivided into two categories. **Static iconics** are static hand postures, such as making a circle by cupping two hands together. **Dynamic iconics** requires the movement of both hands to form the outline of the icon.

Manipulation gestures are used to guide movement in a short feedback loop. Actions are executed as the user is performing the gesture. The difference between manipulation and pantomimic or dynamic iconic is the presence of the feedback loop, where objects "follow" the gestures instead of performing the gesture beforehand and causing a reaction subsequently.

Many researchers have developed prototypes with 3D mid-air gestures. Groenwald et al. used the classification scheme from Aigner et al. to survey existing literature on 3D mid-air gestures. They identified 65 papers that implemented mid-air hand gestures which have primarily been designed for selection, navigation, and manipulation tasks. They did not find any papers that implemented SDAZ. Surprisingly they found that there have been very few evaluations.

As far as I am aware no-one has explored SDAZ with 3D gestures nor conducted a comparison study between different devices and SDAZ which is the focus of the contribution in this project.

Chapter 3

Design

3.1 Platform Choice

Three different platforms were considered for this project. JavaScript with Google Maps JavaScript API, C++ and OpenGL API, and finally Unity game engine.

A rough prototype was implemented in the very early stages of this project using Google Maps JavaScript API and the Leap Motion Controller camera. It was found that the Google Maps JavaScript API lacked the necessary control over the display that this project required. Zooming in the Google Maps JavaScript API was done in discrete steps, and lacked the smooth transition between zoom levels needed for SDAZ.

Cockburn and Savage [3, 11] used C and OpenGL library in their implementation. OpenGL was used for its high frame rate and fluid interactions, which they believed were a setback in Igarashi and Hinckley's [7] Java implementation.

Unity was chosen as the implementation platform. Unity is able to achieve the high performance required for the responsiveness and fluid interactivity Cockburn and Savage [3, 11] sought from OpenGL. In addition to this, Unity is compatible with various platforms and devices such as Windows, Android, and Leap Motion Controller. Unity allowed for a consistency despite working with a variety of platforms and devices. OpenGL served as a backup platform in the event that major issues occurred with Unity. Fortunately this was not the case.

3.2 Gestures Design

Navigating documents require zooming, scrolling and panning. A continuous flow in the document is desired, so manipulation gestures (as classified by Aigner et al. [1]. See Section 2.4), or the presence of a feedback loop, is required for both zooming and moving.

3.2.1 Zooming

For both non-SDAZ and SDAZ the same zooming gestures are used, as there is no reason to have different gestures for the same function (Figure 3.1).

In Groenewald et al. [5] (review of 3D mid-air gestures) zooming utilized semaphoric stroke and dynamic sermaphoric gestures. These gestures are intuitively bi-manual [1]. Naturally these will be combined with a short feedback loop in the system to achieve a manipulation gesture.

Nancel et al. [10] found that two handed, linear gestures had the smallest movement time. Their qualitative results showed that participants preferred two handed and linear

gestures too. With a two handed gesture, users will not be able to scroll and zoom at the same time either. This is consistent with touch techniques, where panning and zooming are determined by whether the user is using one or two fingers, restricting the user to one action at one time.

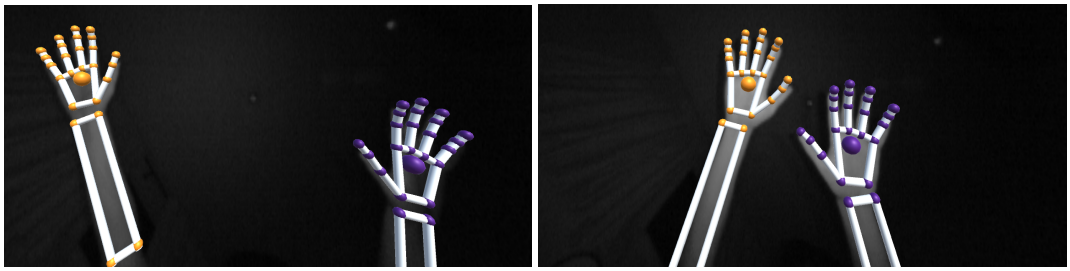


Figure 3.1: Hands moving while performing the zoom gesture as seen from the Leap Motion Controller Visualization Tool. If hands are moved from the positions in the left image to the positions in the right image, a zoom out action would occur. Zoom in for right image to left image positions.

For our design the zooming gesture is two hands open flat pointing downwards. When the hands are moved apart in this position, it means zoom in. It can be thought of the document being stretched out, so that the size (scale) of the document increases. Vice versa when the hands are moved closer together the document is bought back together, or zoomed out.

3.2.2 Movement

For movement SDAZ utilizes panning gestures as SDAZ does not require the user to constantly gesture to move the camera. The user controls the speed rather than the displacement, and once a good speed has been reached, the user will want to remain there until the target is reached or slow down as they approach the target (Figure 3.2).

Non-SDAZ can either use the same panning gesture as SDAZ or use scrolling gestures. Using scrolling gestures is more intuitive for non-SDAZ techniques, whereas using the same panning gestures as SDAZ would maintain consistency not only within the two techniques, but with touch input as well. Users would need to continuously repeat the panning gesture to properly navigate the document, but it is the same situation in touch devices which is standardized and widespread. Therefore consistency will be maintained over intuition and utilize the same panning gestures as SDAZ for non-SDAZ movement also.

Figure 3.2: Hand moving while performing the movement gesture as seen from the Leap Motion Controller Visualization Tool

Although Groenewald et al. [5] found six out of ten panning gestures had users pointing to where they wanted to move, only three had users grabbing and moving. For this design grab and move is required as both the direction and magnitude of movement need to specified for SDAZ. The gesture decided upon has users "grab" or activate panning by forming an "L" shape with their thumb and index finger, representative of a corner of the camera viewpoint. Once activated, the displacement of the hand position from the initial position dictates the displacement or magnitude of the movement.

3.3 SDAZ Algorithm

Similar to Cockburn and Savage [3, 11], a linear relationship between the scroll speed and the zoom level was used over the exponential relationship in Igarashi and Hinckley's [7] implementation.

The algorithm is thus:

$$zoom = 1 - \frac{distFromStart - zoomThreshold}{maxDist - zoomThreshold}$$

Where *zoom* is a value from 0 to 1 (0 minimal possible zoom and 1 maximal possible zoom). *distFromStart* is the magnitude of the displacement (how far the user dragged). *zoomThreshold* is a constant, and indicates how far the user must drag before any sort of zooming occurs. *maxDist* is the maximum possible displacement, calculated off the screen size. Due to us conducting a user experiment of the algorithm over different devices with different screen sizes, the screen size is incorporated as a factor in the algorithm.

$$maxDist = \sqrt{screenHeight^2 + screenWidth^2}$$

A problem with this algorithm occurs when the user begins at a zoom level less than 1 (minimum). The system would first zoom into 1 and then out again. This was solved by checking whether the new zoom value was greater than (more zoomed in) than the current zoom. If it is the case, then the system would remain on the current zoom, and zooming only when the SDAZ algorithm returns a zoom smaller (more zoomed out) than the current zoom level.

In addition to previous work [3, 7, 11], meaning was given to the magnitude of displacement. The distance of the offset represents the distance the document moves per second. In other words, the document will move at rate such that it is equal to if the user was repeating the action once per second on the non-SDAZ technique. For example, the user drags for 100 pixels on the map. In the non-SDAZ technique, the map would move by 100 pixels and stop. In the SDAZ technique, the map would by 100 pixels per second until the user releases.

Users are also able to manually set the zoom level using traditional methods. This provides users a higher level of control as shown in Kratz et al.[9].

Chapter 4

Implementation

4.1 Unity Scene Setup

Due to the many possible (input $\times$ technique $\times$ document) combinations, the translation from user input to screen updates was linearised into a pipeline. The input device component passes data to the technique component, which then passes data to the document component, which controls the actual camera or output.

There are two separate scenes, one for map (Wellington, Porirua, Lower Hutt, Upper Hutt) and one for text (Hamlet[1]). In each scene there is a User `GameObject` which contains a `CoreController` component. The `CoreController` acts as the hub of communications between the input device component, the technique component, and the document component. It is also tasked with dynamically adding these components to the User `GameObject` when the scene is loaded. There is a menu where users can switch input devices, techniques, and documents.

Map Scene

Map images are placed in the scene as sprites. Panning is done by moving the camera parallel to the map images. Zooming is done by moving the camera perpendicular to the map images. There are four layers of map images (Figure 4.1), with differing sizes and levels of detail similar to a mipmap. The layers are spaced in a way such that when the camera zooms out enough for the size of the map displayed on the screen to equal the size of the next smaller one, the smaller one becomes visible instead. This makes sure there is a smooth transition between the different map images and users do not become disorientated by any jumps in display.

Due the maximum size of textures on Android devices, one image per map size was not possible. The map texture for the most detailed map was 26880x18944 pixels in size, far larger than the limit of any Android device. This caused the texture to become extremely compressed to the point of illegibility, as well as taking a significant amount of time to load the application. To circumvent this problem, the map was divided up into 256x256 tiles. They were put together into a single `GameObject` via code and saved as a Unity prefab. Saving as a prefab means the map does not need to be stitched together each time the scene loads, it can simply load in the pre-constructed prefab. This resulted in 10437 files for four layers of maps at varying zoom levels, as opposed to only four, one for each layer. This significantly increased the build time and time taken to change between PC and Android platforms due to the overhead from processing the metadata for 10437 files. But in exchange,

[1] http://shakespeare.mit.edu/hamlet/full.html

the load times and quality of the textures were significantly improved at runtime.

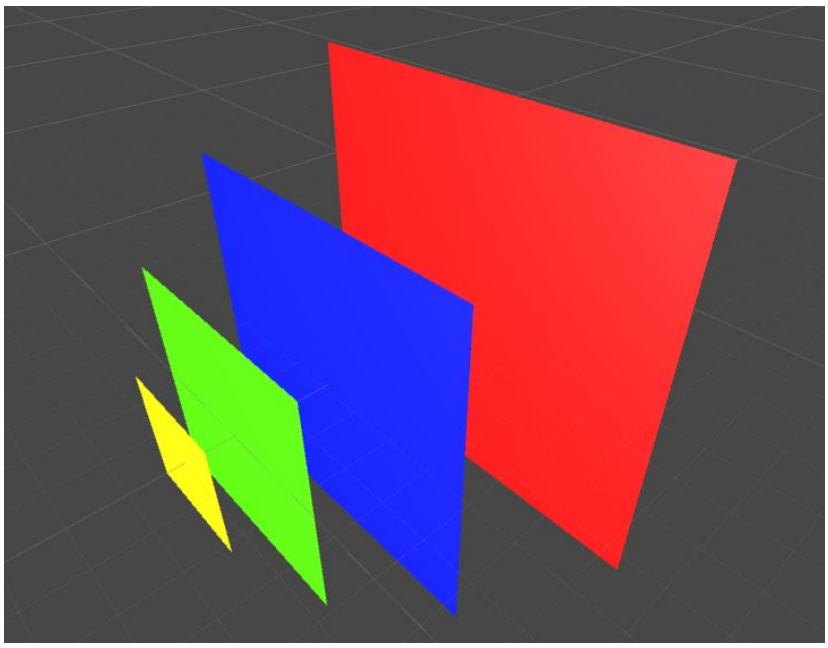


Figure 4.1: Example of map layers. Sizes and spacing exaggerated and coloured for clearer view

Text Scene

The text document scene was implemented mostly through *Unity*'s UI *GameObjects*. The text is rendered through a *Text* UI component. The text displayed is updated via scripts. The entirety of the play is first loaded into a list of strings line by line. The user "position" and zoom level were stored in text, and which lines of the text are updated accordingly. Images in the text document are rendered with the Image UI component, which is also updated by the script. Scrolling is simulated by changing which lines of the document are rendered, and zoom is achieved by changing the font-size. Rich text[2] is used extensively to highlight headings and prevent headings and images from "zooming" by setting lines of text to a fixed font-size. There is also a scrollbar in the text scene, which can be used by mouse.

In the text document scene, several workarounds had to be implemented. *Unity* has no built-in support for rendering large documents, nor images inline with text. Due to the large size of the text (almost 13,000 lines), they could not be all be set into the text UI component. The script therefore looks at the user position, and sets a certain number of lines of the text document starting at that position into the text field of the text UI component. The number of lines set to the UI component is not calculated precisely, but rather given by an overestimate to save computation costs. Due to the high resolution of the screen on the Android device, a large number of lines is needed to be set per frame. This causes significant frame-rate drops when the user is zoomed out to the maximum. Since user inputs are considered per frame, this has a significant impact on the fluidity of user interactions. On SDAZ the document continues to scroll for several moments after the user has disengaged.

The other issue is displaying the images inline with the text. This workaround for this consists of blocks of 20 lines of whitespace in the text for the image to overlay. The image

[2] `https://docs.unity3d.com/Manual/StyledText.html`

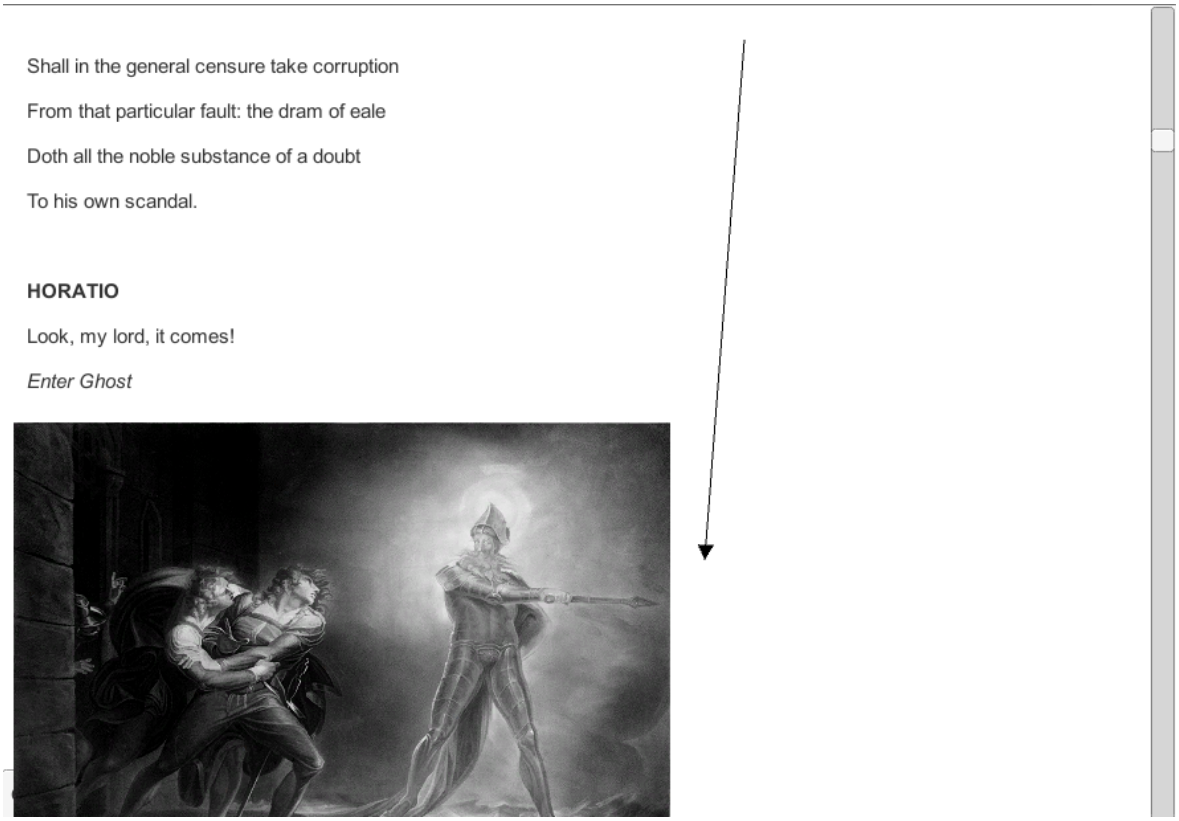


Figure 4.2: Example of text document scene scrolling with SDAZ

is rendered by an Image UI component, whose position is adjusted by script to match the block of whitespace. Again, due to the precise pixel position of lines being not calculated, the position of the image is given by an estimate, and will shift slightly as the user zooms in or out. The line number of the blocks of whitespace is hard-coded.

4.2 InputDevice

The `InputDevice` class reads in raw device data and translates it into user intentions. There are four implementations of this class.

LeapController

The Leap Motion Controller was setup using Unity by importing the LeapCSharp .dll plugin.

The `LeapController` class reads in raw hands and fingers data from the Leap Motion Controller. This raw data contains information such as hand positions, palm directions, and finger directions. The data is then processed to identify the known zoom and move gestures. There is a brief delay of 0.1 seconds in accepting a gesture. We do not want to begin the zooming or moving process because a gesture was detected in passing for a single frame.

Due to a lack of feedback for the user with the Leap Motion Controller, textual feedback is provided. The currently detected gesture (none, zooming, moving) is displayed in the top left corner of the screen and is updated by this class.

TouchInput

Similar to the `LeapController`, the `TouchInput` class reads in touch data. With one finger users can pan, or two fingers for pinching to zoom. The `TouchInput` also has a recognition delay, although shorter than `LeapController` at 0.05 seconds. This is to account for the difference in time between two fingers touching the screen for zooming, as it very unlikely that a user makes contact with the surface of the device with both fingers within the same frame.

MouseInputForMap and MouseInputForText

These two classes are used to process mouse inputs. Users click and drag to pan the document. This class records when the mouse button is held down and how far it is dragged for. There are separate classes for map and text documents due to the difference in controls. In the map, scrolling the mouse wheel means zoom in and zoom out. Whereas in the text document, the wheel scrolls the document up and down by default, and zooms if and only if the left ctrl key is held down.

4.3 Technique

`Technique` takes the user intention provided by the `InputDevice` class and translates it into how the camera should move. The `Technique` class would give the `DocumentController` tasks like move 500 pixels left here, or zoom out by 0.5 there. The `DocumentController` class doing the actual moving by either changing camera position for map or font size and line number for text.

There are two implementations of this class: `NonSDAZ` and `SDAZ`. `NonSDAZ` is insignificant, it passes data directly from `InputDevice` to `DocumentController`. In addition to implementing the algorithm described in Section 3.3, the `SDAZ` class also provides feedback to the user in the form of an arrow. This arrow indicates the start and end of the user's drag, allowing them to see the magnitude and direction of the current movement. This is especially useful for 3D mid-air gestures, as the user's physical positions do not directly translate to screen coordinates.

4.4 DocumentController

Here is where the display is actually modified. The `DocumentController` takes instructions from `Technique` and updates the screen accordingly.

TextController

`TextController` changes the text displayed on screen by changing the line number of the first line displayed. This line number acts as the "position" of the user. The UI textbox in the scene is then fed a certain number of lines starting from the first line displayed. If SDAZ is being used, then the position is updated at a fixed rate in the `FixedUpdate()` function. The `FixedUpdate()` function is a Unity framework function called at regular intervals. It is normally used for physics calculations in games, as the gap between calls do not fluctuate like the `Update()` function which is called per frame. Zooming is done by simply changing the font size of the textbox. Headings are set to maintain a fixed 20pt font unaffected by the zoom.

MapController

The `MapController` modifies the camera position in the map scene to pan and zoom the map, as well as the camera field of view. Map spacing is also calculated and set by this class. The following calculations are required for this.

Camera Field of View (FOV)

The camera FOV is modified so that, at the most zoomed in state, 1 pixel on the screen is equal to 1 pixel on the image. This means across different devices and screen sizes, or even when the window is resized, it is not simply the same information displayed at different sizes. Say when you maximize a window, you expect there to be more information displayed, rather than the same information but larger. The calculation is given by the formula

$$fov = tan^{-1}(\frac{screenHeight}{100 \times 2}) \times 2$$

The screen height is divided by 100 to change pixels into Unity world coordinates. This is then divided by 2 to get a right angle triangle to calculate angles with $tan-1$, this is then multiplied by 2 to get the angle of the whole screen. Note that *screenHeight* is the height of the application window, rather than the physical device height.

Movement and Position Clamping

Panning the map is achieved by moving the camera parallel to the map images. The camera position is clamped at the edges of the map, so that the screen is covered by map data at all times. We need the edge of the map to match up to the edge of the screen at the clamped position. This calculation needs to take into account the zoom level of the camera, the camera field of view, and the aspect ratio. The aspect ratio is used to find the horizontal field of the view of the camera to clamp along the x axis.

Zooming and Map Jumps

Zooming is achieved by moving the camera perpendicular to the map images. As the camera uses a perspective projection moving the camera further/closer to the map effectively zooms it out/in, as the user can see more/less of the map with individual elements on the map becoming smaller/larger.

A jump in camera position needs to be made when transitioning between different maps (see Section 4.1). We need the location displayed on the visible map to remain the same. But due to the size differences of the maps in the scene, these are different coordinates in the scene. The jump is calculated based on the ratio of the sizes of the maps. If the camera was 70% across and 65% down on the original map, then it will be 70% across and 65% down on the new map.

Map Spacing

The maps are spaced out such that if the user was on a z-position (where the z-axis is perpendicular to the maps) the scale of map $m1$ displayed would be equal to the scale of the next map $m2$. Figure 4.3 shows an example of two maps at the same z-position. Both maps are showing the same area at the same scale, but the first image is maximally zoomed in, and the second is more than maximally zoomed out. Note that the second image is impossible during normal use, as the system would have switched automatically to the map in the first image.

4.5 User Interface

A simple user interface is provided for switching between various options for user testing. This UI uses Unity's UI canvas `GameObjects`. Options are changed by simply loading a new

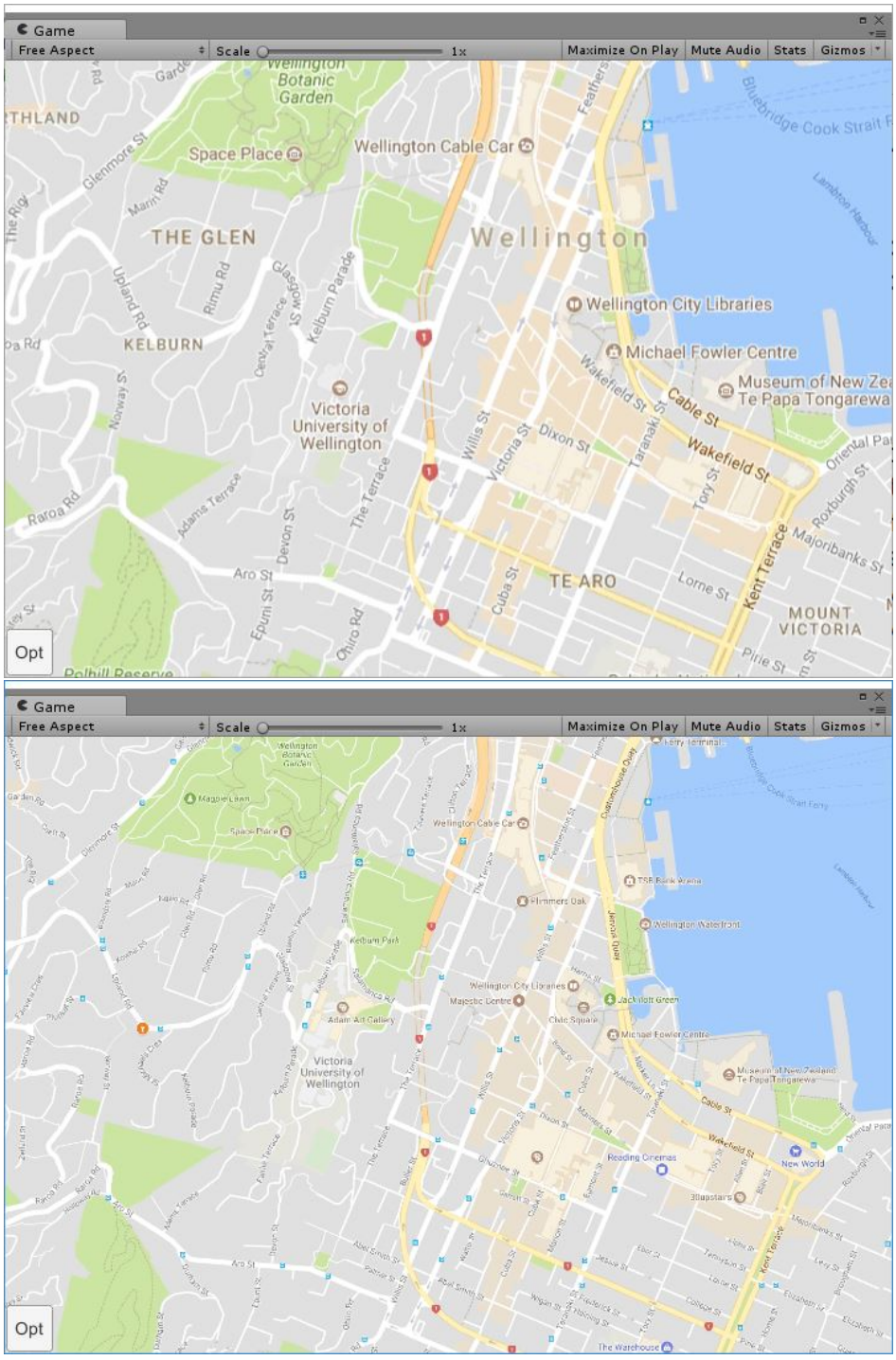


Figure 4.3: Two maps at identical z-positions

scene with the options as parameters. The current settings can also be reset, reducing the load times between the various tasks during the user testing.

Chapter 5

User Study

Figure 5.1: Three different input devices participants used: mouse, touch, and gestures.

This project was approved for user testing by the Human Ethics Committee with Approval number 24679.

To understand the efficiency and effectiveness of SDAZ vs non-SDAZ for mouse, touch, and 3D mid-air gestures a user experiment was conducted. The study replicated the study conducted by Cockburn and Savage [3, 11] for mouse, but extended it to include touch and 3D mid-air gesture input. For the experiment a Windows 10 laptop with a mouse, Samsung Galaxy Tab S3 Android tablet as the touch device, and Leap Motion Controller for 3D mid-air gestures were used.

5.1 Procedure

Participants were first asked to fill in a pre-study survey. Training was then given to explore the different interfaces. They were then asked to perform the tasks which were timed. After all the tasks were completed, they were then asked to fill in a post-study survey. Overall the experiment took between 90 minutes and two hours, depending on the speed at which the participant completed the tasks.

5.2 Participants

30 participants (16 female, 13 male, and 1 other) were recruited for the experiment. 5 participants were under 20 years of age, 15 were 20-24, 7 were 25-29, 1 was in their 30's, and 2 were over 40 (Table 5.1). Almost all participants were very confident with mouse and touch, with all but two participants using mouse and touch daily (Table 5.2, Figure 5.2). Participants were of average confidence with gestures, 25 have never used gestures before, 4

	Male	Female	Other	Total
< 20	2	3	0	5
$20 - 24$	6	8	1	15
$25 - 29$	4	3	0	7
$30 - 39$	0	1	0	1
$40+$	1	1	0	2
Total	13	16	1	30

Table 5.1: Age and gender of participants

	Daily	Weekly	Monthly	Never
Mouse	28	1	1	0
Touch	28	2	0	0
Gestures	0	1	4	25

Table 5.2: Participants' input devicess usage frequency

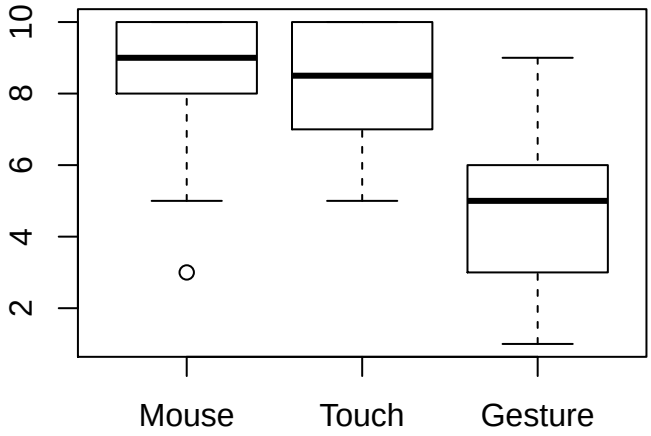


Figure 5.2: Boxplot of participants' confidence with the three input devices.

used it monthly, and 1 participant used it weekly. Each participant received an honorarium for participating in the experiment.

5.3 Background Information Questionnaire

The questionnaire identified participants' previous experience with the technology, as well as general background information. Answers for this questionnaire are summarized in the previous section.

The questions in the questionnaire were as follows:

1. Gender

   Male Female Other Prefer not to say

2. Age

   Up to 20 20-24 25-29 30-39 40+

3. Confidence with mouse and keyboard.
4. Confidence with touch devices.
5. Confidence with 3D midair gestures.
6. Familiarity with a map of Wellington, Lower Hutt and Porirua.
7. Frequency of using mouse and keyboard.
8. Frequency of using touch devices.
9. Frequency of using 3D mid-air gesture recognition devices.
10. Average number of hours spent playing video games every week

Questions 3, 4, 5, and 6 are answered with a scale from 1 to 10. Questions 7, 8, and 9 requires the participant to choose one of daily, weekly, monthly, and never.

5.4 Navigation Tasks

The study conducted was a within subjects experiment where there were different combinations of tasks. There was a total of $3 \times 2 \times 2 = 12$ combinations (input device $\times$ technique $\times$ document type). For each of the 12 combinations there were four task types. This gave a total of 48 tasks each participant performed across all devices. There were four tasks types in total. Two for each document type. Each task type had a short and long version, giving four total tasks per (input device $\times$ technique $\times$ document type) combination. The task types were as follows:

Text Document Locate Picture required participants to locate a specific picture.

Text Document Locate Heading required participants to locate a specific section heading.

Map Locate required participants to locate a landmark known to the participant such as the university campus or the airport, identify it, and roughly center it on the screen.

Map Follow required participants to follow along state highways until they reached a certain destination, identify it, and roughly center it on the screen.

Before each combination (device $\times$ technique) participants were given 5 minutes to familiarize themselves with the combination for both document types. This was a chance to for them to learn the inputs and ask any questions they may have about the interface.

5.4.1 Each task

Each task was read out to the participants at the beginning. Participants could ask for directions back to the starting point or to have the instructions read out again at any time, although the timer was not reset. The time was recorded with a stopwatch. The timing began after the task was read out, and following a three second countdown. The timer was stopped once the objective of the task had been met.

5.4.2 Order of Tasks

All text document tasks were completed before map navigation tasks. This is due to the number of dimensions of each document. Text documents can only go up and down, whereas maps have the added dimension of left and right. By completing text document tasks first, participants are more familiar with the setup, and are "eased into" maps. For each document type, the order of devices and then techniques tested were counter balanced to minimize learning effects. This gave 48 possible orderings. A list of orderings was counter balanced before user studies began for each document type. Participants were assigned an entry from this list depending on when they participated. For example, the first participant was assigned the first ordering in the list, the second participant the second ordering.

5.5 Data Collection

To collect data about the participants they completed a post-survey on what they thought about each (device $\times$ technique) combination. There were two parts to the survey. The first was a NASA Task Load Index (NASA-TLX) assessment [6]. The NASA-TLX is used to measure subjective assessment of workloads in the tasks. The NASA-TLX consists of six workloads with a 21-point scale: mental demand, physical demand, temporal demand, overall performance, effort, and frustration level. The second part of the survey asked the participants what they liked about the combination, and what they did not like about the combination. Participants were asked to rank the six combinations on a likert scale. Participants were videoed performing the experiment to record their interactions and thoughts, and screen captured each input device action.

Mental demand evaluates how hard participants had to think to complete the tasks. Were they simple or complicated?

Physical demand evaluates how much participants had to move to complete the tasks. Were they slack or strenuous?

Temporal demand evaluates how pressured for time the participants were. Were the tasks fast paced of relaxed?

Overall performance evaluates how successful the participants thought they were in performing the tasks. Were they happy with how well they did?

Effort evaluates how hard participants had to work to achieve their level of performance.

Frustration level evaluates how irritated or stressed participants were during the tasks. Were they calm and relaxed as they performed the tasks?

Chapter 6

Results

6.1 Text Document

Table 6.1 are the results of performing a Tukey Honestly Significant Differences test at a 5% significance level fitted by a repeated measures ANOVA test.

$\circ$ indicates the row combination is significantly faster than the column combination

$=$ indicates no significant difference

$\bullet$ indicates the row combination is significantly slower than the column combination

Both Gesture x Non-SDAZ and Gesture x SDAZ are significantly slower than mouse and touch for both SDAZ and non-SDAZ ($p = 0.000$), with no significant difference between SDAZ and non-SDAZ for gestures ($p = 0.999$, truncated). Mouse x SDAZ is significantly slower than Touch x Non-SDAZ ($p = 0.034$), while there are no significant differences between the other pairs ($p = 0.0677, 0.999, 0.417, 0.953, 0.278$, all truncated)

Figure 6.1(a) is a boxplot of the time taken in seconds to complete tasks for text documents. As the outlier on the Gesture x SDAZ combination renders the other plots unreadable. The figure shows that Gesture for both Non-SDAZ and SDAZ are much slower than the other four combinations.

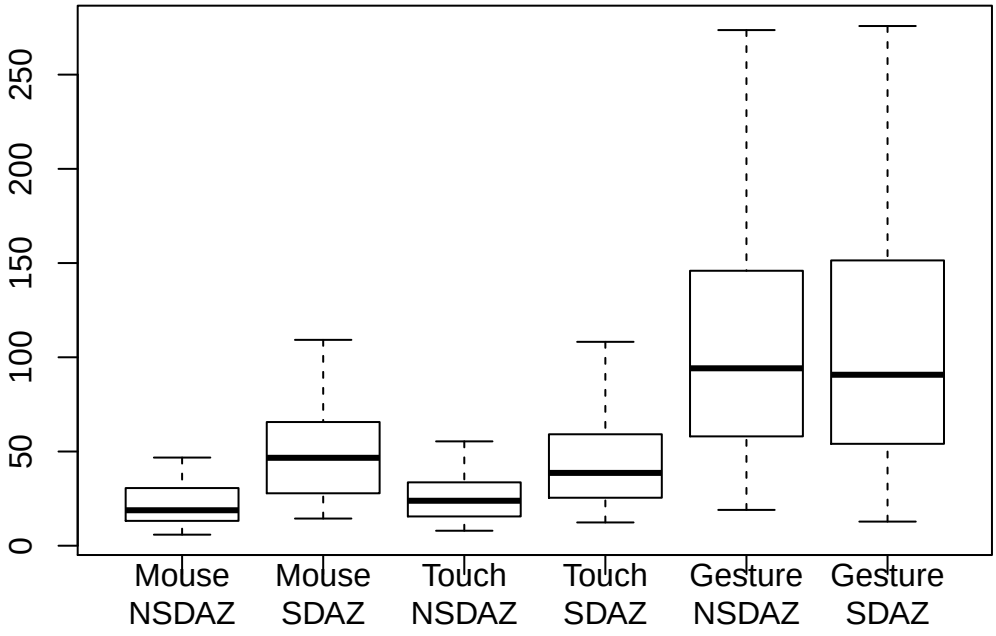
6.2 Map

Table 6.2 are the results of performing a Tukey Honestly Significant Differences test at a 5% significance level fitted by a repeated measures ANOVA test.

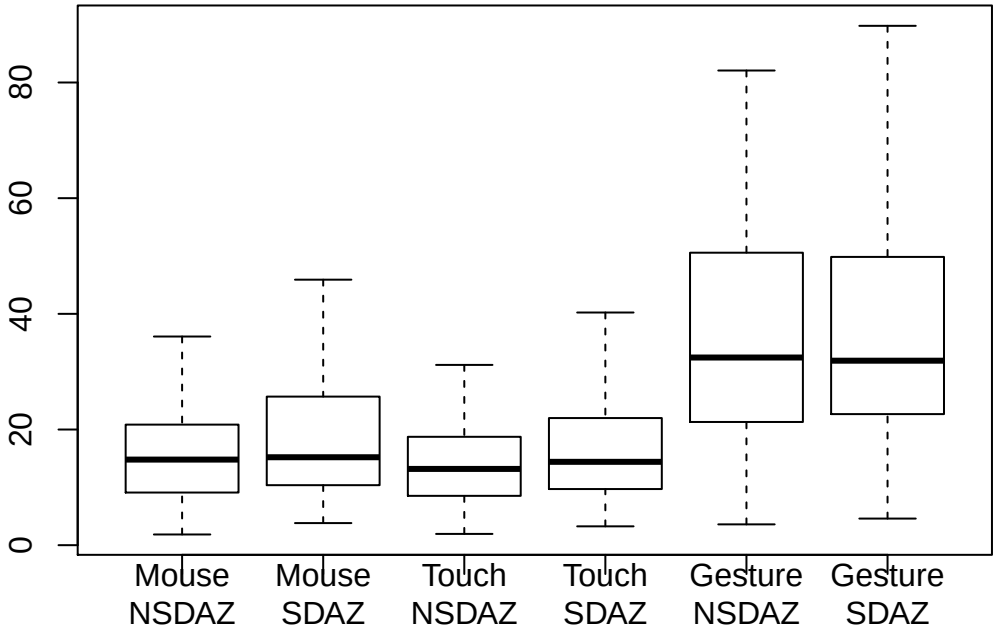
$\circ$ indicates the row combination is significantly faster than the column combination

Combination	Time for Tasks Mean (Std)	Mouse NSDAZ	Mouse SDAZ	Touch NSDAZ	Touch SDAZ	Gesture NSDAZ	Gesture SDAZ
Mouse x NSDAZ	28.29 (31.99)	x	=	=	=	$\circ$	$\circ$
Mouse x SDAZ	53.62 (37.43)	=	x	$\bullet$	=	$\circ$	$\circ$
Touch x NSDAZ	26.05 (13.91)	=	$\circ$	x	=	$\circ$	$\circ$
Touch x SDAZ	45.60 (25.73)	=	=	=	x	$\circ$	$\circ$
Gesture x NSDAZ	23.88 (101.1)	$\bullet$	$\bullet$	$\bullet$	$\bullet$	x	=
Gesture x SDAZ	25.17 (131.56)	$\bullet$	$\bullet$	$\bullet$	$\bullet$	=	x

Table 6.1: Comparisons of combinations for text documents



(a) Text Document.



(b) Map.

Figure 6.1: Boxplot results of combinations. Y Axis time taken (seconds) and X Axis combination type: Mouse Non-SDAZ, Mouse SDAZ, Touch Non-SDAZ, Touch SDAZ, Gesture Non-SDAZ, Gestures SDAZ.

$=$ indicates no significant difference

• indicates the row combination is significantly slower than the column combination.

Combination	Time for Tasks Mean (Std)	Mouse NSDAZ	Mouse SDAZ	Touch NSDAZ	Touch SDAZ	Gesture NSDAZ	Gesture SDAZ
Mouse x NSDAZ	16.00 (9.76)	x	=	=	=	○	○
Mouse x SDAZ	19.21 (12.23)	=	x	=	=	○	○
Touch x NSDAZ	14.57 (9.43)	=	=	x	=	○	○
Touch x SDAZ	17.26 (12.28)	=	=	=	x	○	○
Gesture x NSDAZ	42.85 (33.50)	●	●	●	●	x	=
Gesture x SDAZ	40.16 (26.46)	●	●	●	●	=	x

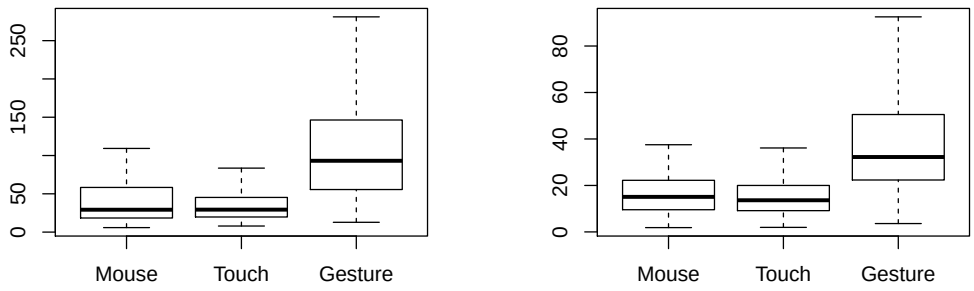
Table 6.2: Comparisons of combinations for maps

Differing from text documents, all the non-gesture combinations have no significant difference.

Figure 6.1(b) is a boxplot of the time taken in seconds to complete tasks for maps. Once again the outliers on the gesture combinations renders the other plots unreadable. The figure shows that Gesture for both Non-SDAZ and SDAZ are much slower than the four other combinations.

6.3 Comparing between Devices

For both text document and map, there are no significant differences between mouse and touch ($p = 0.714, 0.606$, for text and map respectively), with gestures significantly worse ($p = 0.000$) than both mouse and touch. Figure 6.2 is the boxplot of the time taken in seconds to complete tasks, combined by the device type.

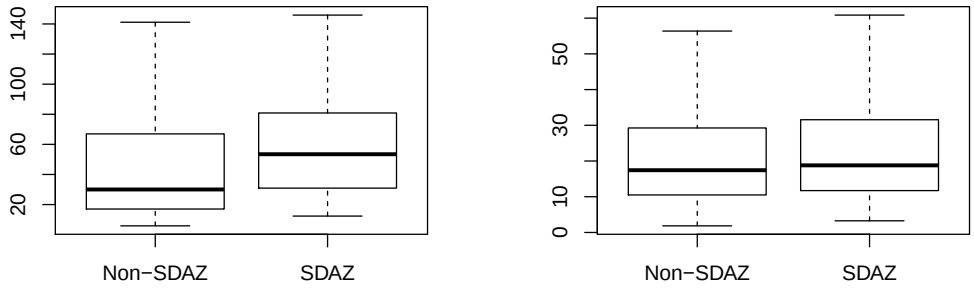


(a) Text Document. (b) Map.

Figure 6.2: Boxplot results of devices. Y Axis time taken (seconds) and X Axis device type: Mouse, Touch, Gesture.

6.4 Comparing between Techniques

When comparing between techniques, however, there is a clear and significant difference between Non-SDAZ and SDAZ ($p = 0.009$) for text document. However there is no such difference for map ($p = 0.513$). Figure 6.3 is the boxplot of the time taken in seconds to complete tasks, combined by the technique.



(a) Text Document. (b) Map.

Figure 6.3: Boxplot results of techniques. Y Axis time taken (seconds) and X Axis technique type: Non-SDAZ, SDAZ.

Combination	Mental Demand	Physical Demand	Temporal Demand	Overall Performance	Effort	Frustration
Mouse x NSDAZ	4.467	3.617	6.817	5.817	5.033	4.717
Mouse x SDAZ	5.267	4.517	6.958	5.620	5.208	4.360
Touch x NSDAZ	4.550	6.887	7.493	5.287	6.993	5.217
Touch x SDAZ	4.927	4.527	6.620	6.120	5.073	4.990
Gesture x NSDAZ	11.603	16.500	11.155	10.621	15.562	13.779
Gesture x SDAZ	12.350	14.583	10.417	11.340	12.723	11.497

Table 6.3: Mean NASA-TLX scores for each combination. A lower score is better.

6.5 NASA Task Load Index Results

Table 6.3 gives the mean scores of the NASA-TLX survey. A lower score is better. Figure 6.4 presents this data as a bar graph. Each bar in a subgroup represents a combination in the same order as previous result tables. The subgroups represent, in order, mental demand, physical demand, temporal demand, overall performance, effort, and frustration.

As shown on the graph, the two gesture combinations scored much worse than the non-gesture combinations. Therefore the following list of significant differences will assume gesture combinations are significantly worse than non-gesture combinations unless explicitly stated. For mental demand there are no significant differences. For physical demand Touch x Non-SDAZ is significantly worse than Mouse x Non-SDAZ. For temporal demand Gestures x SDAZ is not significantly different from any of the non-gesture combinations $(p = 0.0762, 0.0993, 0.239, 0.051)$, while Gesture x Non-SDAZ is only not significantly worse than Touch x Non-SDAZ $(p = 0.072)$. For overall performance, effort, and frustration there were no significant differences at a 5% significance level.

Figure 6.5 shows the mean rankings of each combination. Mouse x Non-SDAZ had a mean ranking of 2.150. Mouse x SDAZ had a mean ranking of 2.050. Touch x Non-SDAZ had a mean ranking of 3.300. Touch x SDAZ had a mean ranking of 3.167. Gesture x Non-SDAZ had a mean ranking of 5.533. Gesture x SDAZ had a mean ranking of 4.933. A smaller number means higher rank (number 1 is best, number 6 is worst).

Overall mouse combinations were ranked significantly better than touch combinations, which were significantly better than gesture combinations. There were no significant differences within devices. But SDAZ had a higher mean rank for every device which indicates it was subjectively more effective.

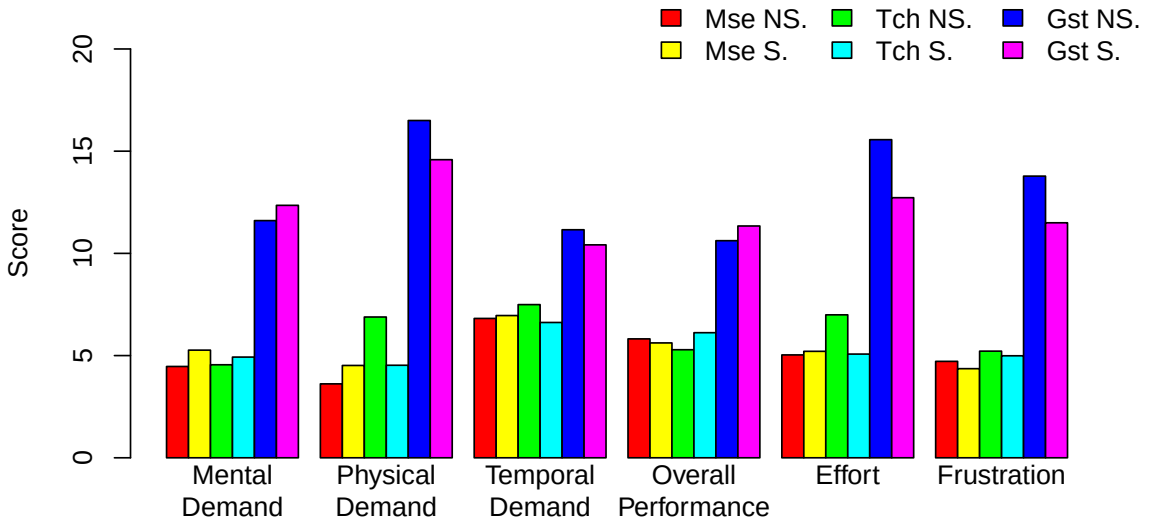


Figure 6.4: Means scores of NASA-TLX responses for the different combinations. A smaller number means higher rank (1 is best and 6 is worst).

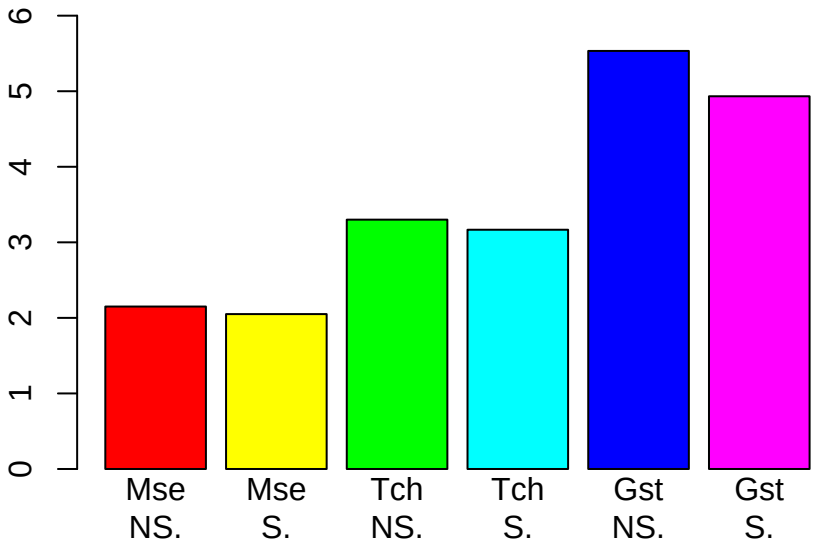


Figure 6.5: Bar graph of mean rankings of NASA-TLX responses. Mouse ranked best, then touch, followed by 3D gestures.

Chapter 7

Discussion

7.1 Differences in Results

Previous work [3, 11] found that using SDAZ significantly improved users' time taken to complete tasks. In this study, it was found that participants were actually slower using SDAZ. This difference can be attributed to the differences in participant pool and the year.

All of Cockburn and Savage's [3, 11] participants were graduate level computer science students, who can be expected to be familiar with the purposes of SDAZ and therefore adapt quickly. This study recruited students from across the university, from first-year undergraduates to PhD students. A few participants were not even students. The students came from a range of fields as well, including: literature, language, health, biology, and finance. Not all of whom can be expected to be familiar with the purposes of a navigation technique.

Cockburn and Savage's study was conducted in 2002, when computers did not have the ubiquity they do today. People today are familiar with the standard navigation techniques in web browsers and text browsers. All but one participant in our study were familiar with Google Maps. Compared to 2002, Cockburn and Savage were using a graphics editor to simulate navigating a map. Due to the widespread use and standardization of navigation techniques today, the participants can be reasonably expected to perform faster with them due to their familiarity with the behaviour.

7.2 Comparing between Techniques

Section 6.4 showed there was a discrepancy between Non-SDAZ and SDAZ for text documents but not for map. This is possibly due to the difference in the size of the documents and the distances participants were required to travel.

As SDAZ requires users to drag across the screen, the size of the screen affects the maximum scroll speed. In text document tasks, users were required to travel over long distances, often simply holding down the drag as the document moves. This was not a problem in Non-SDAZ, where the scrolling speed is directly related to how fast participants moved physically.

This was not a problem for map tasks. As the distances travelled were much shorter than in text documents, participants did not need to drag across the entire length of the screen to reach the destination. So although the limitation of the physical screen size still existed, it did not affect the results.

7.3 3D mid-air Gestures

3D mid-air gestures performed significantly worse across the experiment. There are a number of issues with gestures that resulted in a low score for the evaluation. These issues could potentially be solved with more feedback from the system or with more training.

7.3.1 Range

The speed at which SDAZ techniques move is dependent on the length of the user's drag. Although the Leap Motion Controller has a large detection range $\tilde{8}$0cm above the table surface, only 2 or 3 participants took advantage of this fact. Almost all participants remained within 20 or 30 centimeters of the table surface. This resulted in extremely short drags, and therefore extremely slow movement in the document. Theoretically, Gesture x SDAZ should outperform Mouse x SDAZ and Touch x SDAZ, as the latter two both have the physical limitation of screen space for dragging. Gesture x SDAZ is able to achieve a much higher maximum speed because of this. Additional training or feedback from the system ensuring participants' awareness of this fact should be implemented. With further training, users should be aware of, and use to their advantage, the detection range of the Leap Motion Controller. Feedback regarding the participants' hand position relative to the range of detection of the Leap Motion Controller would also help the more observant participants, letting them know that they are not utilizing the maximum capabilities of the device.

7.3.2 Gesture Design

Another major issue is the participants' ability to perform the correct gesture. The implementation of the system had a strict requirement on the pose, with various aspects to consider such as angle between fingers or having fingers horizontal on a plane parallel to the table. Many participants struggled to maintain this strict posture, and therefore were unable to maintain the drag for Gesture x SDAZ. Despite feedback in the form of an arrow on the screen, some participants would not realize that their drag had been interrupted and reset, and would continue moving their hand downwards, further adding to the range problem discussed previously. Many participants commented on the physical stress of gestures, which also reflected in the NASA-TLX results. Around three or four participants, after struggling with the gestures for a period of time, lost focus. They began waving their hands around out of desperation, without any regard of actually performing the gesture. This would go on for about 20-30 seconds before they refocused and attempted the gesture again.

7.4 Zooming

A common trend across the slower participants (in terms of total time, across all tasks) was not zooming. Zooming out allows participants to scroll faster. The majority of participants zoomed out before moving, especially for long distance tasks. The slower participants would not do this, they would stay on the maximum zoom (which tasks start at) and scroll towards the target at the slower speed. This is a problem more related to the participants' abilities with computers in general. As participants were recruited from all around the university rather than only computer science students, not all participants had high productivity with computers. SDAZ assists novice users in this example, by zooming out and increasing the movement speed automatically. Although this is an isolated example, SDAZ is not suitable for novice users in general.

7.5 Page Skipping

In the text document tasks, common feedback was the lack of page skipping options. Participants were only allowed to use the scrollbar for Mouse x Non-SDAZ, which acts as a baseline. The scrollbar allowed participants to jump to any part of the document. There was no such functionality available for the five other combinations, forcing participants to manually scroll through the document. Although this feature did not significantly increase participants' performance, as users often "overshot" when using the scrollbar. Due to the large size of the document, small movements on the scrollbar still results in large movements on the document. This is the biggest problem SDAZ is addressing.

7.6 Refining SDAZ

Other than the time taken to perform text document tasks, SDAZ was not significantly different to Non-SDAZ. Even ranking higher, although not significantly, than Non-SDAZ for each device. With further refinements to SDAZ parameters such as scrolling speed or zoom levels [4], SDAZ could outperform non-SDAZ.

7.7 3D Space

This experiment only concerned 1D and 2D spaces. 3D space would be more suitable for 3D mid-air gestures input. Example applications include globe browsers [2] (Google Earth, NASA WorldWind) or 3D modeling programs (Maya, Blender).

7.8 Usability and Calibration of 3D Mid-air Gestures

As a large number of participants struggled with performing the gestures to the strict standard required, more flexible detection or a calibration phase may improve the usability of the gestures. Calibration would allow the system to give more leeway to any possible idiosyncrasies of individual participants.

7.9 Limitations

There were some limitations. 30 participants were recruited for this study, but if there had been significantly more participants (e.g $n > 100$) this may have yielded different results. The study was conducted in a lab setting which had limited time to test the different combinations (up to 2 hours in total). Participants were new to using 3D gestures and were more familiar with interacting with a mouse. Studying these techniques over a longitudinal time period may provide different insights.

English was not the first language of a few participants, and they sometimes had trouble understanding the task. This was an issue in the map follow task for maps, where participants were required to react to instructions in real time. Furthermore, instructions were given in cardinal directions, which even some native speakers could not follow immediately.

7.10 Future Work

This is a summary of the previous sections in this chapter. This study can be improved by:

- Increased number of participants
- Increased feedback of information to users using 3D mid-air gestures
- Slacker requirements on gestures
- Increased training for participants
- Page-skipping function for touch and 3D mid-air gestures
- Refining SDAZ parameters
- Exploring 3D space

Chapter 8

Conclusion

Speed-Dependent Automatic Zooming (SDAZ) is a document navigation technique first developed by Igarashi and Hinckley. The idea of SDAZ is for the document to zoom out automatically as the user's scrolling speed increases, allowing for the visual flow of information across the screen to remain constant. Previous formal evaluations using mouse showed that navigation tasks were solved significantly faster using SDAZ [3, 11]. Subjective preferences also favoured SDAZ.

Our study extended the previous work in SDAZ by evaluating SDAZ for touch and 3D mid-air gesture input devices in addition to mouse and keyboard. I found the time taken for participants to perform tasks did not significantly differ between SDAZ and Non-SDAZ techniques within devices, with room for SDAZ speeds to improve. 3D mid-air gestures performed poorly compared to mouse and touch input devices. I believe this is due to participants' unfamiliarity with the input device. With further training and more usable systems in the form of additional feedback, gestures could potentially be able to perform at a standard equal to mouse or touch. Subjective feedback indicated participants preferred SDAZ for 3D gesture input, opening the possibility that SDAZ could be standardized for gesture input devices.

Bibliography

[1] AIGNER, R., WIGDOR, D., BENKO, H., HALLER, M., LINDBAUER, D., ION, A., ZHAO, S., AND KOH, J. Understanding mid-air hand gestures: A study of human preferences in usage of gesture types for hci. *Microsoft Research TechReport MSR-TR-2012-111 2* (2012).

[2] COCKBURN, A., LOOSER, J., AND SAVAGE, J. Around the world in seconds with speed-dependent automatic zooming. 35–36.

[3] COCKBURN, A., AND SAVAGE, J. Comparing speed-dependent automatic zooming with traditional scroll, pan and zoom methods. In *Proceedings of BritishHCI* (2004), Springer, pp. 87–102.

[4] COCKBURN, A., SAVAGE, J., AND WALLACE, A. Tuning and testing scrolling interfaces that automatically zoom. In *Proceedings of CHI* (2005), ACM, pp. 71–80.

[5] GROENEWALD, C., ANSLOW, C., ISLAM, J., ROONEY, C., PASSMORE, P., AND WONG, W. Understanding 3D mid-air hand gestures with interactive surfaces and displays: A systematic literature review. In *Proceedings of BritishHCI* (2016).

[6] HART, S. G., AND STAVELAND, L. E. Development of nasa-tlx (task load index): Results of empirical and theoretical research. *Advances in psychology 52* (1988), 139–183.

[7] IGARASHI, T., AND HINCKLEY, K. Speed-dependent automatic zooming for browsing large documents. In *Proceedings of UIST* (2000), ACM, pp. 139–148.

[8] KARAM, M., AND SCHRAEFEL, M. C. A taxonomy of gestures in human computer interactions. Tech. Rep. 261149, University of Southhampton, 2005.

[9] KRATZ, S., BRODIEN, I., AND ROHS, M. Semi-automatic zooming for mobile map navigation. In *Proceedings of MobileHCI* (2010), ACM, pp. 63–72.

[10] NANCEL, M., WAGNER, J., PIETRIGA, E., CHAPUIS, O., AND MACKAY, W. Mid-air pan-and-zoom on wall-sized displays. In *Proceedings of CHI* (2011), ACM, pp. 177–186.

[11] SAVAGE, J. Speed-dependent automatic zooming. *Honours Thesis, University of Canterbury* (2002).

[12] WOBBROCK, J. O., MORRIS, M. R., AND WILSON, A. D. User-defined gestures for surface computing. In *Proceedings of CHI* (2009), ACM, pp. 1083–1092.

[13] ZHAI, S., SMITH, B. A., AND SELKER, T. Improving browsing performance: A study of four input devices for scrolling and pointing tasks. In *Proceedings of INTERACT* (1997), Springer, pp. 286–293.

Appendices

Appendix A

Human Ethics Committee Approval

TE WHARE WĀNANGA O TE ŪPOKO O TE IKA A MĀUI
VICTORIA
UNIVERSITY OF WELLINGTON

Phone 0-4-463 5480
Email susan.corbett@vuw.ac.nz

MEMORANDUM

TO	Jiaheng Wang
COPY TO	Craig Anslow
FROM	AProf Susan Corbett, Convener, Human Ethics Committee

DATE	13 June 2017
PAGES	1

SUBJECT	**Ethics Approval: 24679** Understanding SDAZ - Mouse vs. Touch vs. 3D Gestures

Thank you for your application for ethical approval, which has now been considered by the Standing Committee of the Human Ethics Committee.

Your application has been approved from the above date and this approval continues until 30 November 2017. If your data collection is not completed by this date you should apply to the Human Ethics Committee for an extension to this approval.

Best wishes with the research.

Kind regards

Susan Corbett

Convener, Victoria University Human Ethics Committee